1. a. Include the Metadata element in Homepage.html for providing description as "IEKart's is an online shopping website that sells goods in retail. This company deals with various categories like Electronics, Clothing, Accessories etc.

   b. Enhance the Homepage.html of IEKart's Shopping Application by adding appropriate sectioning elements.

   c. Make use of appropriate grouping elements such as list items to "About Us" page of IEKart's Shopping Application.

   d. Link "Login", "SignUp" and "Track order" to "Login.html", "SignUp.html" and "Track.html" page respectively. Bookmark each category to its details of IEKart's Shopping application.

   e. Add the © symbol in the Home page footer of IEKart's Shopping application.

   f. Add the global attributes such as contenteditable, spellcheck, id etc. to enhance the Signup Page functionality of IEKart's Shopping application.

homepage.html

```
<!DOCTYPE html>
<html>
    <head>
        <Title>
IEKart’s Online Shopping
         </title>
        <meta description="IEKart's is an online shopping website that sells goods in retail.
This company deals with various categories like Electronics, Clothing, Accessories etc.">
    </head>
    <body>
<header>
<nav>
<a href="login.html">Login</a> &nbsp;|
<a href="registration.html">SignUp</a> &nbsp; |
<a href="track.html">Track order</a>
</nav>
</header>
<main>
<section>
<article>
        <center><h1>Welcome to IEKart's online shopping website</h1></center>
</article>
```

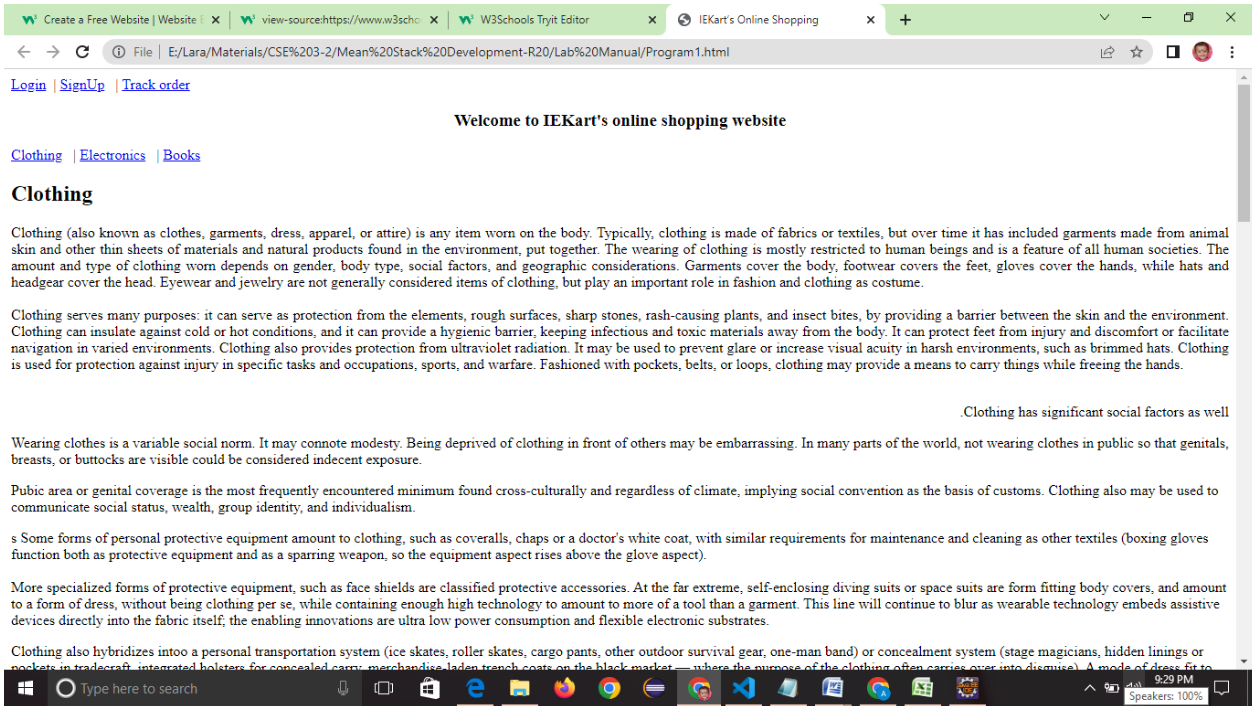
```html
<article>
<nav>
<a href="#clothing">Clothing</a> &nbsp; |
<a href="#electronics">Electronics</a> &nbsp; |
<a href="#books">Books</a>
</nav>
</article>
</section>
<section id="clothing">
<h2>Clothing</h2>
<p align="justify">Clothing (also known as clothes, garments, dress, apparel, or attire) is
any item worn on the body. Typically, clothing is made of fabrics or textiles, but over
time it has included garments made from animal skin and other thin sheets of materials
and natural products found in the environment, put together. The wearing of clothing is
mostly restricted to human beings and is a feature of all human societies. The amount and
type of clothing worn depends on gender, body type, social factors, and geographic
considerations. Garments cover the body, footwear covers the feet, gloves cover the
hands, while hats and headgear cover the head. Eyewear and jewelry are not generally
considered items of clothing, but play an important role in fashion and clothing as
costume.<br><br>

Clothing serves many purposes: it can serve as protection from the elements, rough
surfaces, sharp stones, rash-causing plants, and insect bites, by providing a barrier
between the skin and the environment. Clothing can insulate against cold or hot
conditions, and it can provide a hygienic barrier, keeping infectious and toxic materials
away from the body. It can protect feet from injury and discomfort or facilitate navigation
in varied environments. Clothing also provides protection from ultraviolet radiation. It
may be used to prevent glare or increase visual acuity in harsh environments, such as
brimmed hats. Clothing is used for protection against injury in specific tasks and
occupations, sports, and warfare. Fashioned with pockets, belts, or loops, clothing may
provide a means to carry things while freeing the hands.<br><br>

<p dir="rtl">Clothing has significant social factors as well. </p>
<p align="justify">Wearing clothes is a variable social norm. It may connote modesty.
Being deprived of clothing in front of others may be embarrassing. In many parts of the
world, not wearing clothes in public so that genitals, breasts, or buttocks are visible could
be considered indecent exposure. </p>
<p contenteditable="true">Pubic area or genital coverage is the most frequently
encountered minimum found cross-culturally and regardless of climate, implying social
convention as the basis of customs. Clothing also may be used to communicate social
status, wealth, group identity, and individualism.</p>s
```

```
Some forms of personal protective equipment amount to clothing, such as coveralls,
chaps or a doctor's white coat, with similar requirements for maintenance and cleaning as
other textiles (boxing gloves function both as protective equipment and as a sparring
weapon, so the equipment aspect rises above the glove aspect).<br><br> More
specialized forms of protective equipment, such as face shields are classified protective
accessories. At the far extreme, self-enclosing diving suits or space suits are form fitting
body covers, and amount to a form of dress, without being clothing per se, while
containing enough high technology to amount to more of a tool than a garment. This line
will continue to blur as wearable technology embeds assistive devices directly into the
fabric itself; the enabling innovations are ultra low power consumption and flexible
electronic substrates.

<p>Clothing also hybridizes intoo a personal transportation system (ice skates, roller
skates, cargo pants, other outdoor survival gear, one-man band) or concealment system
(stage magicians, hidden linings or pockets in tradecraft, integrated holsters for concealed
carry, merchandise-laden trench coats on the black market — where the purpose of the
clothing often carries over into disguise). A mode of dress fit to purpose, whether stylistic
or functional, is known as an outfit or ensemble.Hussain's very own one-stop solution for
all your shopping needs !</p>
<img src="E:\Lara\Materials\CSE 3-2\Mean Stack Development-R20\Lab
Manual\clothing1.jpg" border="3" style="border-color:yellow;" height="250"
width="400">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
<img src="E:\Lara\Materials\CSE 3-2\Mean Stack Development-R20\Lab
Manual\clothing2.jpg" border="3" style="border-color:pink;" height="250"
width="400"><br>
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
<img src="E:\Lara\Materials\CSE 3-2\Mean Stack Development-R20\Lab
Manual\clothing3.jpg" border="3" style="border-color:blue;"  height="700"
width="450"><br>
</section>
<section id="electronics">
<h2>Electronics </h2>
<table border="1" cellspacing="5">
<tr>
<td>
<center><img src="charger.jpg" height="150" width="200"></center><br>
<center><pre>VIVO CHARGER ORIGINAL FAST
CHARGER For Vivo Y21 / Vivo Y 21
Charger Original Mobile Charger Wall Charger,
Android Smartphone Charger
Certified Heavey Duty Charger
```

```
Hi Speed Fast Charging Travel Charger
With 1.2 Meter Micro USB Charging
Data Cable VVO (2.4 Ampere , ZK8, White)</pre></center>
<br><center><b>MRP&nbsp;&nbsp;&dollar;378</b></center>
</td>
<td>
<center><img src="wallcharger.jpg" height="150" width="200"></center><br>
<center><pre>cablebasket 3 in 1 Charging Triple
USB Cable Fast Charging Multi Pins for
Android iPhone and Type C Mobile Charger</pre></center>
<br><center><b>MRP&nbsp;&nbsp;&dollar;299</b> </center>
</td>
<td>
<center><img src="multiplugadopter.jpg" height="150" width="200"></center><br>
<center><pre>Varkaus 3 Pin Universal Multi Plug Adaptor
| Travel Adaptor with Surge Protector |
Universal Socket | Extension Cord (Pack of 1, Pearl)</pre></center>
<br><center><b>MRP&nbsp;&nbsp;&dollar;149</b> </center>
</td>
</tr>
</table>
<p>Dont believe us? Click on the offers and check it out for yourself !!</p>
</section>
<section id="books">
<h2>Books </h2>
<table border="1" cellspacing="5">
<tr>
<td>
<center><img src="book1.jpg" height="250" width="200"></center><br>
<center><pre>
Fear Not: Be Strong
by Swami Tathagatananda (Author)
</pre></center>
<br><center><b>MRP&nbsp;&nbsp;&dollar;30</b></center>
</td>
<td>
<center><img src="book2.jpg" height="250" width="200"></center><br>
<center><pre>
Here, There and Everywhere:
Best-Loved Stories of
Sudha Murty</pre></center>
<br><center><b>MRP&nbsp;&nbsp;&dollar;135</b> </center>
```

```
</td>
<td>
<center><img src="book3.jpg" height="250" width="200"></center><br>
<center><pre>
Life’s Amazing Secrets: How to
Find Balance and Purpose in Your
Life | Inspirational Zen book
on motivation, self-development &
healthy living</pre></center>
<br><center><b>MRP&nbsp;&nbsp;&dollar;135</b> </center>
</td>
<td>
<center><img src="book4.jpg" height="200" width="200"></center><br>
<center><pre>
NAVDURGAA Satyam 176 Pages A4
Size Notebook - Single Line
Ruled (29 x 21 cm) | Ruled notebooks
for Writing | Register Notebook
for Students | Pack Set of 6 With
Free 5 Ballpens for Smooth Writing
</pre></center>
<br><center><b>MRP&nbsp;&nbsp;&dollar;378</b></center>
</td>
<td>
<center><img src="book5.jpg" height="250" width="250"></center><br>
<center><pre>
IKIGAI, ATOMIC HABITS, PSYCHOLOGY
OF MONEY AND THE SUBTLE
ART OF NOT GIVING A F*CK
(COMBO BOOKS)
</pre></center>
<br><center><b>MRP&nbsp;&nbsp;&dollar;670</b></center>
</td>

</tr>
</table>
</section>
<br>
<br>
<footer>
<a href="aboutus.html" target="_blank">About Us</a> &nbsp;|&nbsp;
<a href="privacypolicy.html">Privacy Policy</a> &nbsp; | &nbsp;
```

```
&phone;<a href="tel:+91 9999999999">Contact Us</a> &nbsp; | &nbsp;
<a href="freqask.html">FAQ</a> &nbsp; | &nbsp;
<a href="termsandcond.html">Terms & Conditions</a> <br>
Copyrights reserved &copy; 2018 | Giri <br>
Like and Connect with us at <a
href="mailto:manager@iekart.com">manager@iekart.com</a>
</footer>
</boDY>
</html>
```



Login | SignUp | Track order

Welcome to IEKart's online shopping website

Clothing | Electronics | Books

Clothing

Clothing (also known as clothes, garments, dress, apparel, or attire) is any item worn on the body. Typically, clothing is made of fabrics or textiles, but over time it has included garments made from animal skin and other thin sheets of materials and natural products found in the environment, put together. The wearing of clothing is mostly restricted to human beings and is a feature of all human societies. The amount and type of clothing worn depends on gender, body type, social factors, and geographic considerations. Garments cover the body, footwear covers the feet, gloves cover the hands, while hats and headgear cover the head. Eyewear and jewelry are not generally considered items of clothing, but play an important role in fashion and clothing as costume.

Clothing serves many purposes: it can serve as protection from the elements, rough surfaces, sharp stones, rash-causing plants, and insect bites, by providing a barrier between the skin and the environment. Clothing can insulate against cold or hot conditions, and it can provide a hygienic barrier, keeping infectious and toxic materials away from the body. It can protect feet from injury and discomfort or facilitate navigation in varied environments. Clothing also provides protection from ultraviolet radiation. It may be used to prevent glare or increase visual acuity in harsh environments, such as brimmed hats. Clothing is used for protection against injury in specific tasks and occupations, sports, and warfare. Fashioned with pockets, belts, or loops, clothing may provide a means to carry things while freeing the hands.

.Clothing has significant social factors as well

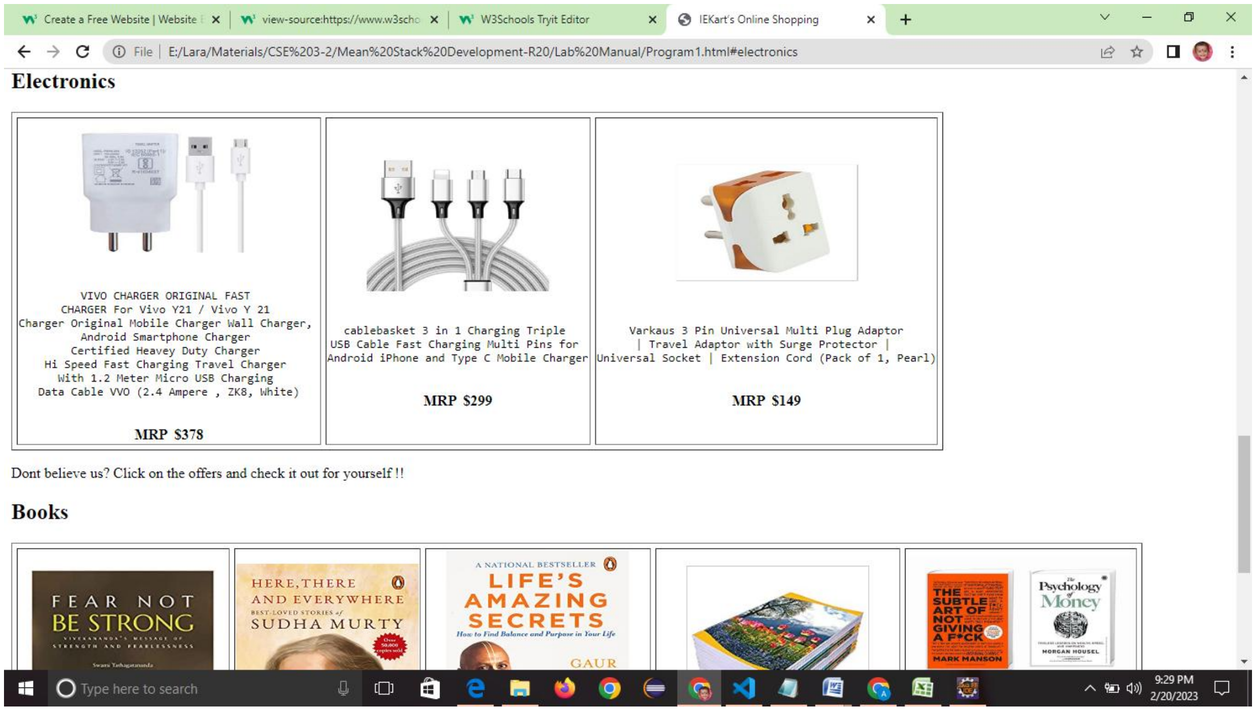
Wearing clothes is a variable social norm. It may connote modesty. Being deprived of clothing in front of others may be embarrassing. In many parts of the world, not wearing clothes in public so that genitals, breasts, or buttocks are visible could be considered indecent exposure.

Pubic area or genital coverage is the most frequently encountered minimum found cross-culturally and regardless of climate, implying social convention as the basis of customs. Clothing also may be used to communicate social status, wealth, group identity, and individualism.

s Some forms of personal protective equipment amount to clothing, such as coveralls, chaps or a doctor's white coat, with similar requirements for maintenance and cleaning as other textiles (boxing gloves function both as protective equipment and as a sparring weapon, so the equipment aspect rises above the glove aspect).

More specialized forms of protective equipment, such as face shields are classified protective accessories. At the far extreme, self-enclosing diving suits or space suits are form fitting body covers, and amount to a form of dress, without being clothing per se, while containing enough high technology to amount to more of a tool than a garment. This line will continue to blur as wearable technology embeds assistive devices directly into the fabric itself; the enabling innovations are ultra low power consumption and flexible electronic substrates.

Clothing also hybridizes intoo a personal transportation system (ice skates, roller skates, cargo pants, other outdoor survival gear, one-man band) or concealment system (stage magicians, hidden linings or



Electronics

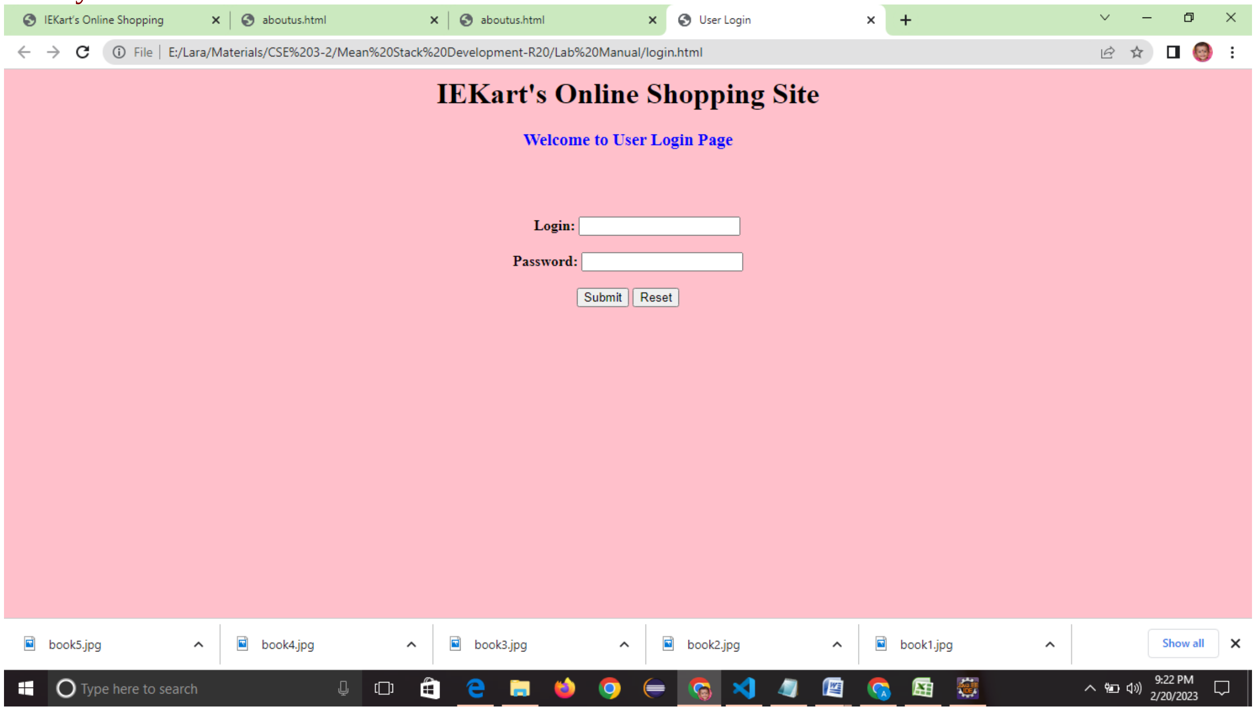
VIVO CHARGER ORIGINAL FAST CHARGER For Vivo Y21 / Vivo Y 21 Charger Original Mobile Charger Wall Charger, Android Smartphone Charger Certified Heavey Duty Charger Hi Speed Fast Charging Travel Charger With 1.2 Meter Micro USB Charging Data Cable VVO (2.4 Ampere , ZK8, White) **MRP $378**	cablebasket 3 in 1 Charging Triple USB Cable Fast Charging Multi Pins for Android iPhone and Type C Mobile Charger **MRP $299**	Varkaus 3 Pin Universal Multi Plug Adaptor | Travel Adaptor with Surge Protector | Universal Socket | Extension Cord (Pack of 1, Pearl) **MRP $149**

Dont believe us? Click on the offers and check it out for yourself !!

Books

login.html

```
<html>
<head>
<title> User Login </title>
</head>
<body bgcolor="pink">

   <h1><center>IEKart's Online Shopping Site</center></h1>
<h3><font color="blue"><center>Welcome to User Login
Page</center></font></h3><br><br><br>
   <center>
<form method="post"> &nbsp;&nbsp;&nbsp;&nbsp;
<b>Login: <input type="text" name="login"/> <br><br>
Password: <input type="password" name="password"/> <br><br>
<input type="submit" value="Submit"/>
<input type="reset" value="Reset"/>
</b>
</form>
</center>
</body> </html>
```


IEKart's Online Shopping
aboutus.html
aboutus.html
User Login
File | E:/Lara/Materials/CSE%203-2/Mean%20Stack%20Development-R20/Lab%20Manual/login.html

IEKart's Online Shopping Site

Welcome to User Login Page

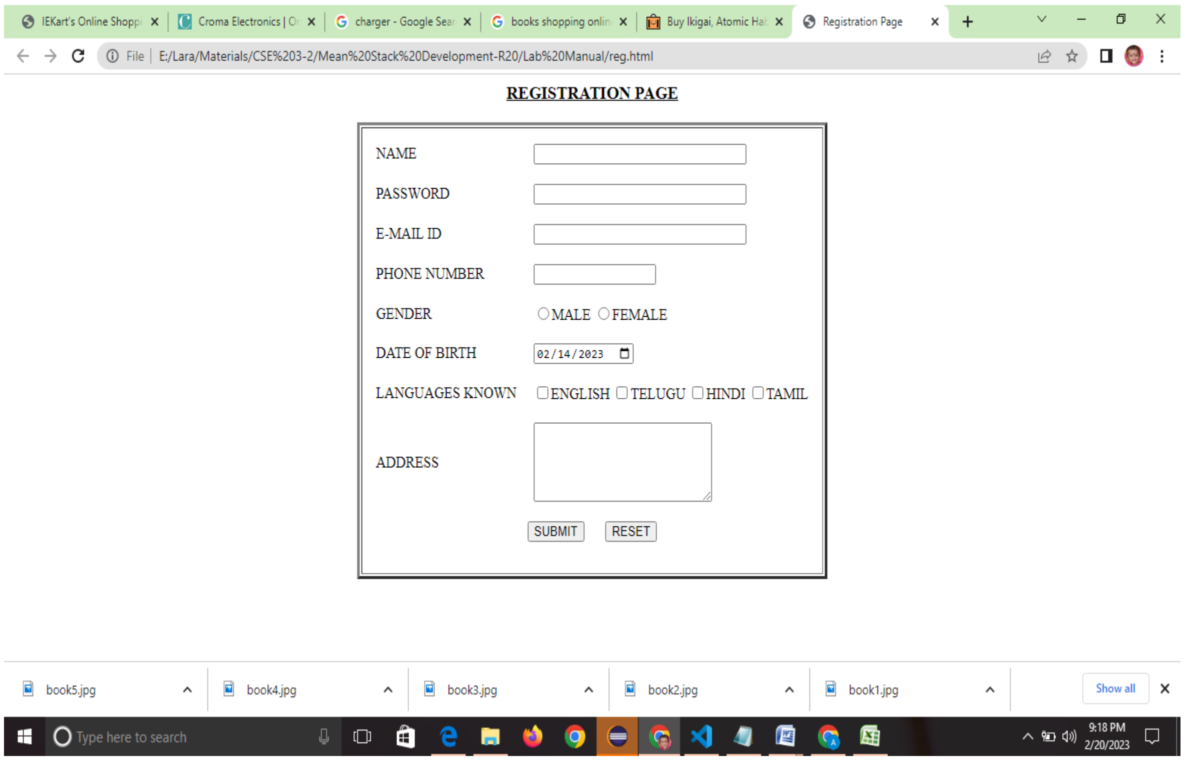
Login:

Password:

registration.html

```
<html>
```

```
<head>
<title>Registration Page</title>
</head>
<body>
<center>
<h3 align="center"><u>REGISTRATION PAGE</u></h3>
<table border="3">
<tr><td>
<form name="f1" action="" method="post" onsubmit="">
<table cellspacing="10" cellpadding="5">
<tr><td>NAME</td><td><input type="text" size="30" name="uname"/></td></tr>
<tr><td>PASSWORD</td><td><input type="password" size="30"
name="pass"/></td></tr>
<tr><td>E-MAIL ID</td><td><input type="text" size="30" name="email"/></td></tr>
<tr><td> PHONE NUMBER</td><td><input type="text" size="15"
name="phone"/></td></tr>
<tr><td>GENDER</td><td><input type="radio" name="gen" value="m" />MALE
<input type="radio" name="gen" value="f" />FEMALE </td></tr>
<tr><td>DATE OF BIRTH</td>
<td>
   <input type="date">
</td></tr>
<tr><td>LANGUAGES KNOWN</td>
<td>
<input type="checkbox" value="eng" name="lang" />ENGLISH
<input type="checkbox" value="tel" name="lang" />TELUGU
<input type="checkbox" value="hin" name="lang" />HINDI
<input type="checkbox" value="tam" name="lang" />TAMIL
</td></tr>
<tr>
<td>ADDRESS</td>
<td><textarea name="addr" cols="25" rows="5"></textarea></td></tr>
<tr><td colspan="2" align="center"><input type="submit"
value="SUBMIT"/>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<input type="reset"
value="RESET" /></td>
</tr>
</table>
</form>
</td></tr></table>
</center>
</body>
</html>
```

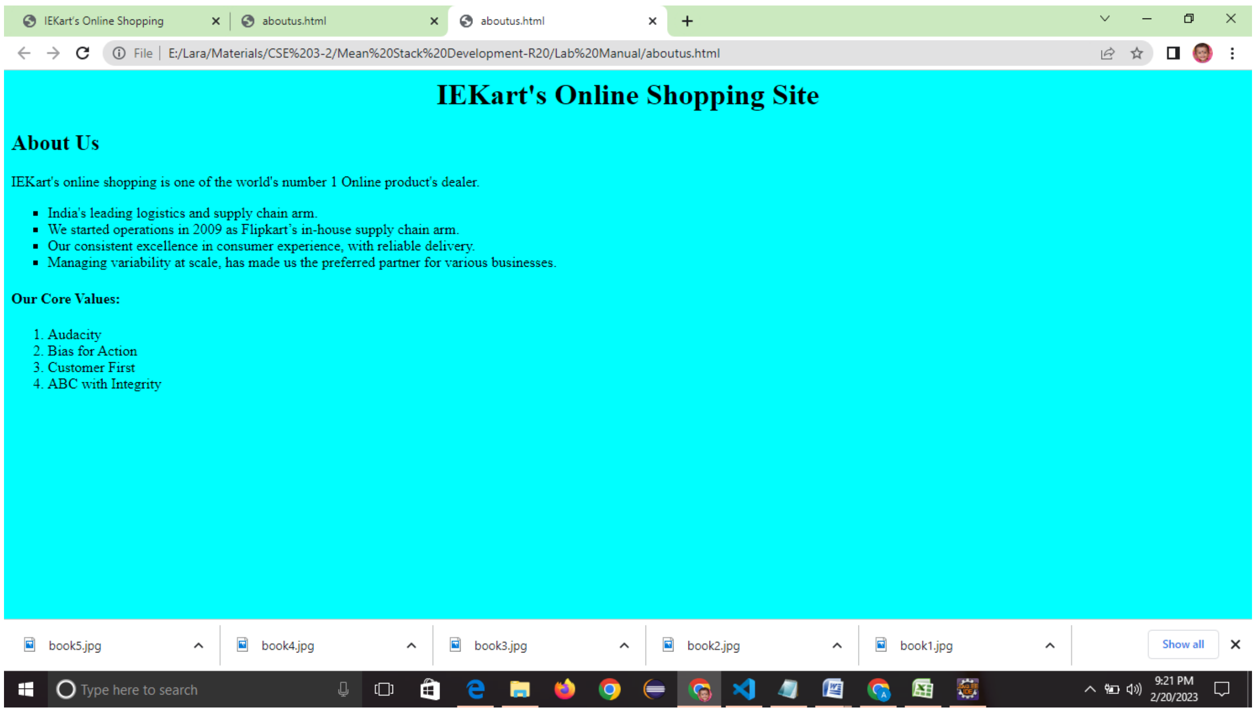


REGISTRATION PAGE

NAME	
PASSWORD	
E-MAIL ID	
PHONE NUMBER	
GENDER	○MALE ○FEMALE
DATE OF BIRTH	02/14/2023
LANGUAGES KNOWN	☐ENGLISH ☐TELUGU ☐HINDI ☐TAMIL
ADDRESS	
	SUBMIT RESET

aboutus.html

```
<html>
    <body bgcolor="aqua">
        <h1><center>IEKart's Online Shopping Site</center></h1>
        <h2>About Us</h2>
IEKart's online shopping is one of the world's number 1 Online product's dealer.
<ul style="list-style-type:square">
<li>India's leading logistics and supply chain arm.</li>
<li>We started operations in 2009 as Flipkart’s in-house supply chain arm.</li>
<li>Our consistent excellence in consumer experience, with reliable delivery.</li>
<li>Managing variability at scale, has made us the preferred partner for various businesses.</li>
</ul>
<h4>Our Core Values:</h4>
<ol>
<li>Audacity</li>
<li>Bias for Action</li>
<li>Customer First</li>
<li>ABC with Integrity</li>
</ol>

    </body>
</html>
```



IEKart's Online Shopping Site

About Us

IEKart's online shopping is one of the world's number 1 Online product's dealer.

- India's leading logistics and supply chain arm.
- We started operations in 2009 as Flipkart's in-house supply chain arm.
- Our consistent excellence in consumer experience, with reliable delivery.
- Managing variability at scale, has made us the preferred partner for various businesses.

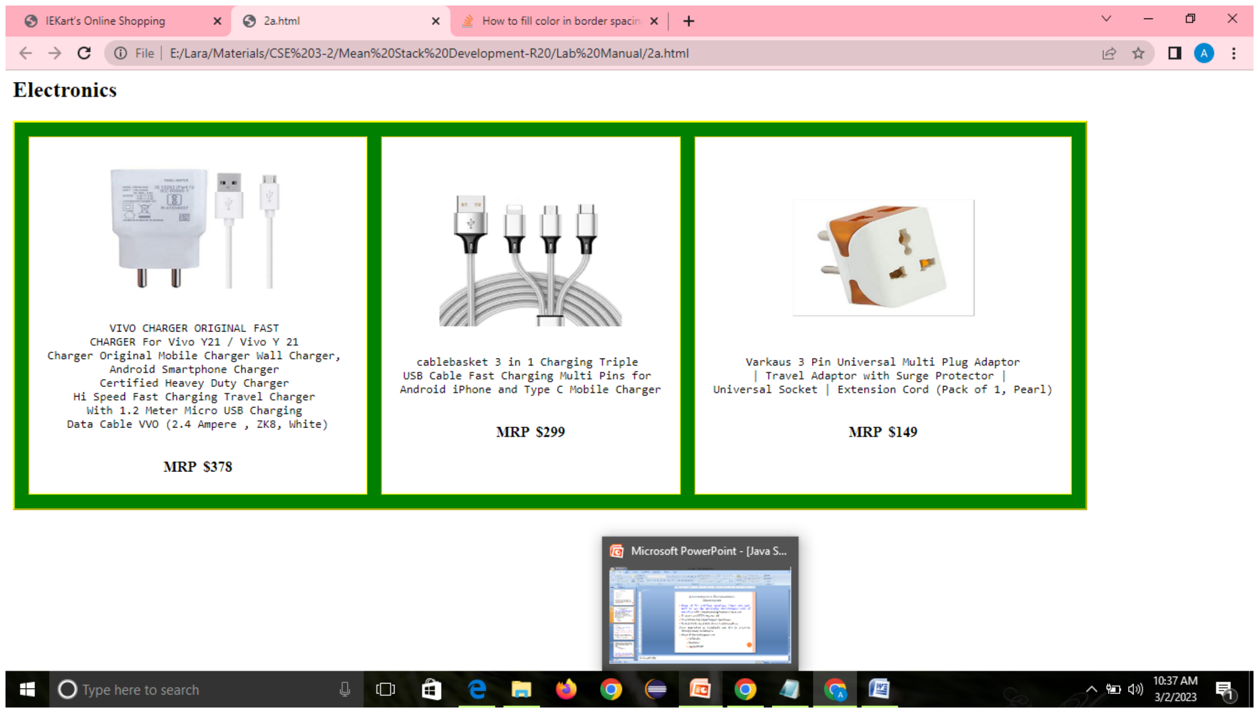
Our Core Values:

1. Audacity
2. Bias for Action
3. Customer First
4. ABC with Integrity

2. A. Enhance the details page of IEKart's Shopping application by adding a table element to display the available mobile/any inventories.

```
<html>
<body>
<h2>Electronics </h2>
<table border="2" style="border-color:yellow;background:green;" cellspacing="15"
cellpadding="20">
<tr>
<td style="background-color: white">
<center><img src="charger.jpg" height="150" width="200"></center><br>
<center><pre>VIVO CHARGER ORIGINAL FAST
CHARGER For Vivo Y21 / Vivo Y 21
Charger Original Mobile Charger Wall Charger,
Android Smartphone Charger
Certified Heavey Duty Charger
Hi Speed Fast Charging Travel Charger
With 1.2 Meter Micro USB Charging
Data Cable VVO (2.4 Ampere , ZK8, White)</pre></center>
<br><center><b>MRP&nbsp;&nbsp;&dollar;378</b></center>
</td>
<td style="background-color: white">
<center><img src="wallcharger.jpg" height="150" width="200"></center><br>
<center><pre>cablebasket 3 in 1 Charging Triple
```

```
USB Cable Fast Charging Multi Pins for
Android iPhone and Type C Mobile Charger</pre></center>
<br><center><b>MRP&nbsp;&nbsp;&dollar;299</b> </center>
</td>
<td style="background-color: white">
<center><img src="multiplugadopter.jpg" height="150" width="200"></center><br>
<center><pre>Varkaus 3 Pin Universal Multi Plug Adaptor
| Travel Adaptor with Surge Protector |
Universal Socket | Extension Cord (Pack of 1, Pearl)</pre></center>
<br><center><b>MRP&nbsp;&nbsp;&dollar;149</b> </center>
</td>
</tr>
</table>
</body>
</html>
```



IEKart's Online Shopping
2a.html
How to fill color in border spacin
File | E:/Lara/Materials/CSE%203-2/Mean%20Stack%20Development-R20/Lab%20Manual/2a.html

Electronics

```
VIVO CHARGER ORIGINAL FAST
CHARGER For Vivo Y21 / Vivo Y 21
Charger Original Mobile Charger Wall Charger,
Android Smartphone Charger
Certified Heavey Duty Charger
Hi Speed Fast Charging Travel Charger
With 1.2 Meter Micro USB Charging
Data Cable VVO (2.4 Ampere , ZK8, White)
```

MRP $378

```
cablebasket 3 in 1 Charging Triple
USB Cable Fast Charging Multi Pins for
Android iPhone and Type C Mobile Charger
```

MRP $299

```
Varkaus 3 Pin Universal Multi Plug Adaptor
| Travel Adaptor with Surge Protector |
Universal Socket | Extension Cord (Pack of 1, Pearl)
```

MRP $149

2b. Using the form elements create Signup page for IEKart's Shopping application.

2c. Enhance Signup page functionality of IEKart's Shopping application by adding attributes to input elements.

```
<html>

<head>

<title>Registration Page</title>
```

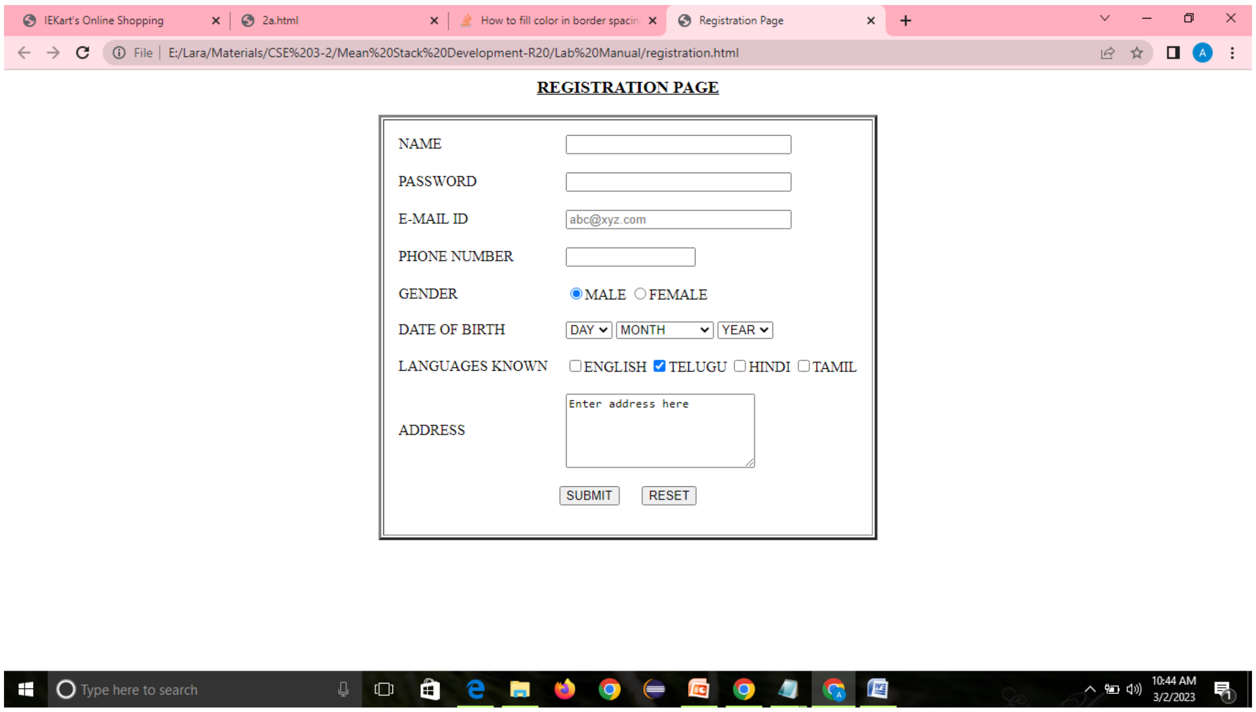
```
</head>

<body>

<center>

<h3 align="center"><u>REGISTRATION PAGE</u></h3>

<table border="3">

<tr><td>

<form name="f1" action="" method="post" onsubmit="">

<table cellspacing="10" cellpadding="5">

<tr><td>NAME</td><td><input type="text" size="30" name="uname" required/></td></tr>

<tr><td>PASSWORD</td><td><input type="password" size="30"

name="pass"/></td></tr>

<tr><td>E-MAIL ID</td><td><input type="text" size="30" name="email"
placeholder="abc@xyz.com" required/></td></tr>

<tr><td> PHONE NUMBER</td><td><input type="text" size="15"

name="phone"/></td></tr>

<tr><td>GENDER</td><td><input type="radio" name="gen" value="m" checked/>MALE

<input type="radio" name="gen" value="f" />FEMALE </td></tr>

<tr><td>DATE OF BIRTH</td>

<td><select name="day">

<option value="day">DAY</option>

<option value="1">1</option>

<option value="2">2</option>

<option value="3">3</option>

<option value="4">4</option>

<option value="5">5</option>

<option value="6">6</option>

<option value="7">7</option>
```

```
<option value="8">8</option>

<option value="9">9</option>

<option value="10">10</option>

<option value="11">11</option>

<option value="12">12</option>

<option value="13">13</option>

<option value="14">14</option>

<option value="15">15</option>

<option value="16">16</option>

<option value="17">17</option>

<option value="18">18</option>

<option value="19">19</option>

<option value="20">20</option>

<option value="21">21</option>

<option value="22">22</option>

<option value="23">23</option>

<option value="24">24</option>

<option value="25">25</option>

<option value="26">26</option>

<option value="27">27</option>

<option value="28">28</option>

<option value="29">29</option>

<option value="30">30</option>

<option value="31">31</option>

</select>

<select name="month">
```

```
<option value="month">MONTH</option>

<option value="jan">JANUARY</option>

<option value="feb">FEBRUARY</option>

<option value="mar">MARCH</option>

<option value="apr">APRIL</option>

<option value="may">MAY</option>

<option value="jun">JUNE</option>

<option value="jul">JULY</option>

<option value="aug">AUGUST</option>

<option value="sep">SEPTEMBER</option>

<option value="oct">OCTOBER</option>

<option value="nov">NOVEMBER</option>

<option value="dec">DECEMBER</option>

</select>

<select name="year">

<option value="year">YEAR</option>

<option value="1986">1986</option>

<option value="1987">1987</option>

<option value="1988">1988</option>

<option value="1989">1989</option>

<option value="1990">1990</option>

<option value="1991">1991</option>

<option value="1992">1992</option>

<option value="1993">1993</option>

<option value="1994">1994</option>

<option value="1995">1995</option>
```

```
<option value="1996">1996</option>

<option value="1997">1997</option>

<option value="1998">1998</option>

<option value="1999">1999</option>

<option value="2000">2000</option>

<option value="2001">2001</option>

<option value="2002">2002</option>

<option value="2003">2003</option>

<option value="2004">2004</option>

<option value="2005">2005</option>

<option value="2006">2006</option>

<option value="2007">2007</option>

<option value="2008">2008</option>

<option value="2009">2009</option>

<option value="2010">2010</option>

<option value="2011">1980</option>

<option value="2012">1981</option>

<option value="2013">1982</option>

<option value="2014">1983</option>

</select></td></tr>

<tr><td>LANGUAGES KNOWN</td>

<td>

<input type="checkbox" value="eng" name="lang" />ENGLISH

<input type="checkbox" value="tel" name="lang" checked/>TELUGU

<input type="checkbox" value="hin" name="lang" />HINDI

<input type="checkbox" value="tam" name="lang" />TAMIL
```

```
</td></tr>
<tr>
<td>ADDRESS</td>
<td><textarea name="addr" cols="25" rows="5">Enter address here</textarea></td></tr>
<tr><td colspan="2" align="center"><input type="submit"
value="SUBMIT"/>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<input type="reset"
value="RESET" /></td>
</tr>
</table>
</form>
</td></tr></table>
</center>
</body>
</html>
```


IEKart's Online Shopping
2a.html
How to fill color in border spacin
Registration Page
File | E:/Lara/Materials/CSE%203-2/Mean%20Stack%20Development-R20/Lab%20Manual/registration.html
Type here to search
10:44 AM
3/2/2023

REGISTRATION PAGE

NAME	
PASSWORD	
E-MAIL ID	abc@xyz.com
PHONE NUMBER	
GENDER	● MALE ○ FEMALE
DATE OF BIRTH	DAY MONTH YEAR
LANGUAGES KNOWN	☐ ENGLISH ☑ TELUGU ☐ HINDI ☐ TAMIL
ADDRESS	Enter address here
SUBMIT RESET	

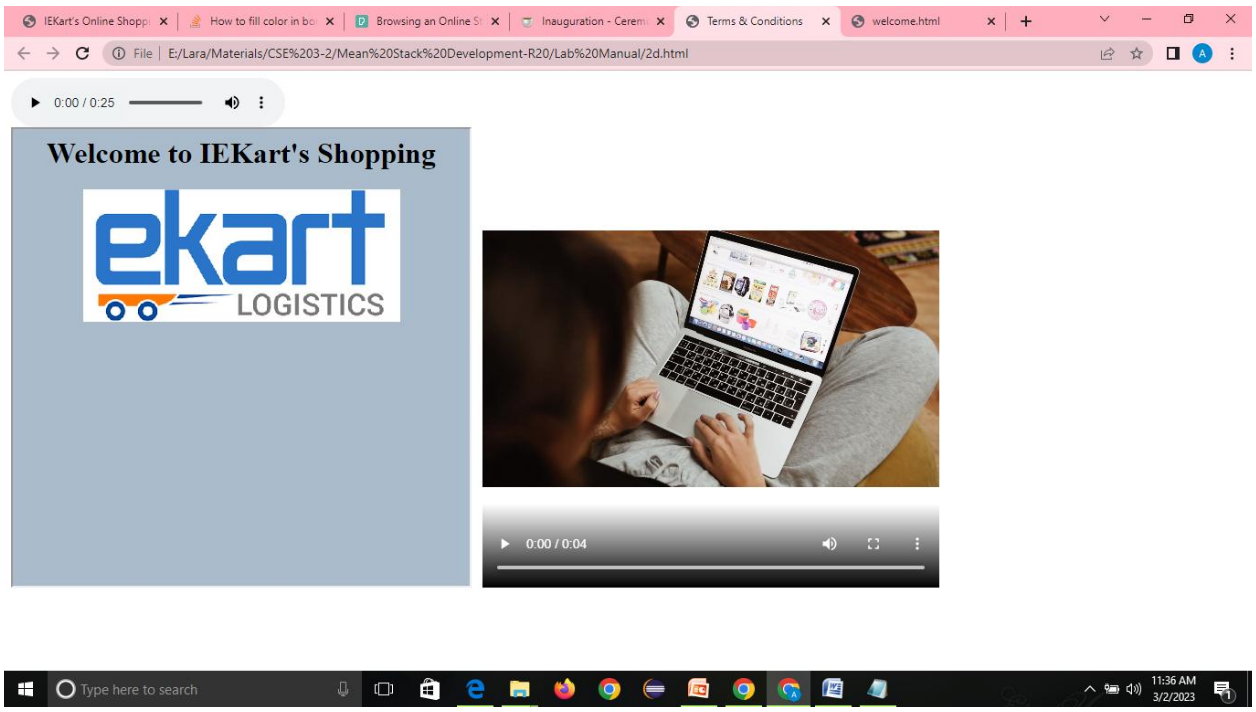
2d. Add media content in a frame using audio, video, iframe elements to the Home page of IEKart's Shopping application.

2d.html

```
<html>

<head>

<title>Terms & Conditions</title>

</head>

<body>

<audio src="E:\Lara\Materials\CSE 3-2\Mean Stack Development-R20\Lab Manual\bgm.mp3"
controls autoplay>

</audio>

<br>

<iframe src="welcome.html" height="500" width="500">

</iframe>

&nbsp;&nbsp;<video src="E:\Lara\Materials\CSE 3-2\Mean Stack Development-R20\Lab
Manual\videoclip.mp4" height="500" width="500" controls autoplay>

</video>

</body>

</html>
```

welcome.html

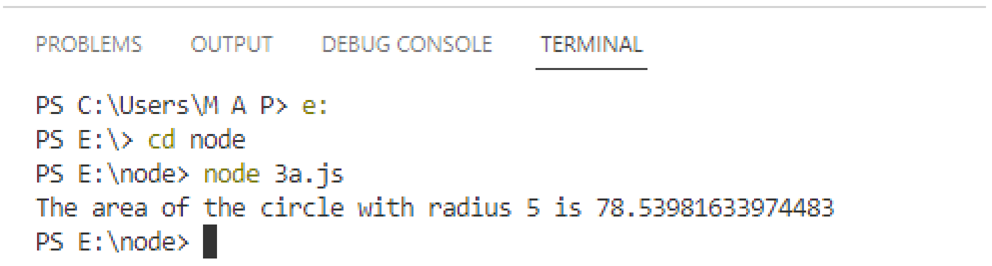
```
<html>

<body bgcolor="#abc">

<center><h1>Welcome to IEKart's Shopping<h1>

<img src="E:\Lara\Materials\CSE 3-2\Mean Stack Development-R20\Lab
Manual\download.png"></img>

</center>

</body>

</html>
```



3a. Write a JavaScript program to find the area of a circle using radius (var and let - reassign and observe the difference with var and let) and PI (const)

```
// Define the radius and Pi
var radius = 5;
const Pi = Math.PI;

// Calculate the area of the circle using the formula A = Pi * r^2
const area = Pi * Math.pow(radius, 2);

// Print the result
console.log(`The area of the circle with radius ${radius} is ${area}`);
```

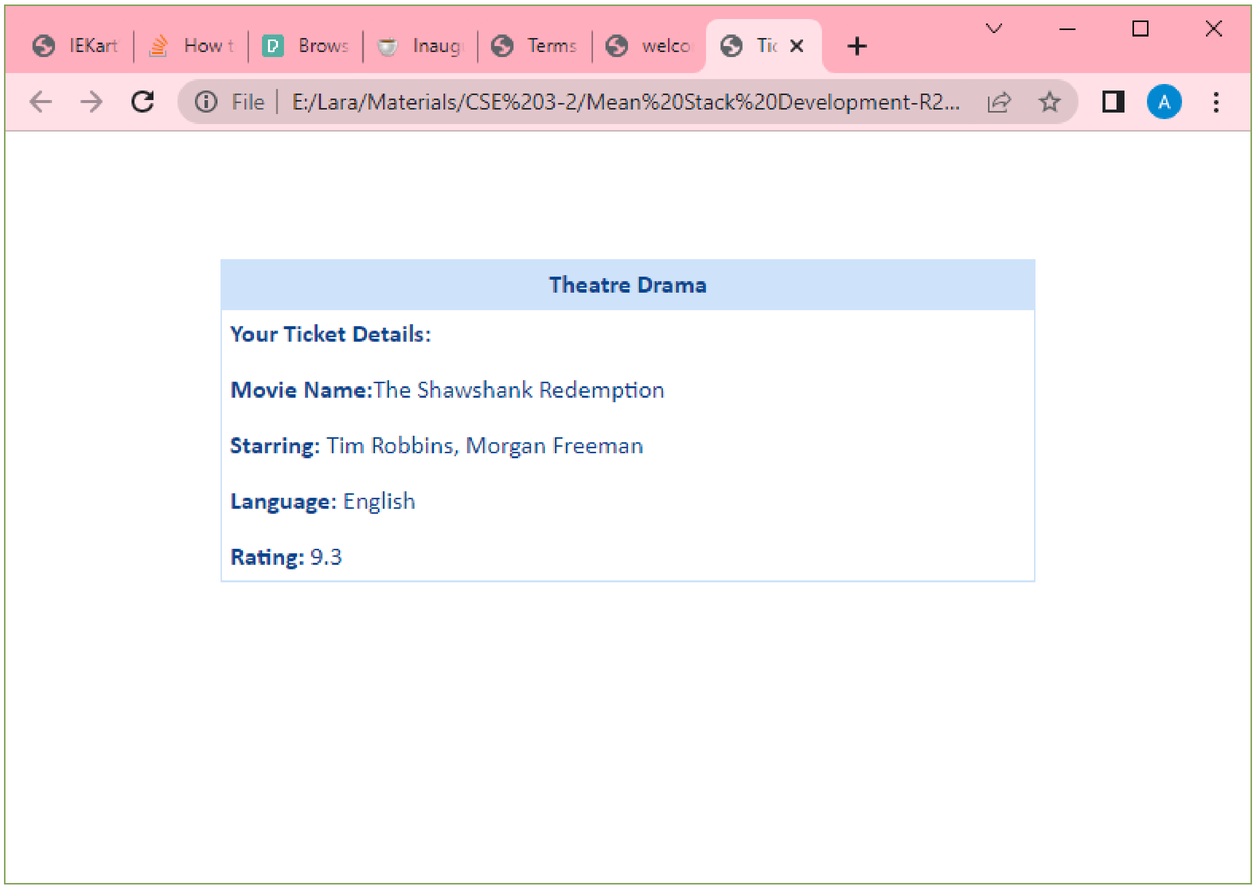
Output:



```
PS C:\Users\M A P> e:
PS E:\> cd node
PS E:\node> node 3a.js
The area of the circle with radius 5 is 78.53981633974483
PS E:\node> 
```

3b. Write JavaScript code to display the movie details such as movie name, starring, language, and ratings. Initialize the variables with values of appropriate types. Use template literals wherever necessary.

```
<!DOCTYPE html>

<html>


<head>

    <title>Ticket Details</title>

    <style>

        div#maincontent {

            height: 160px;

            width: 500px;

            border: 1px solid #CEE2FA;

            text-align: left;

            color: #08438E;

            font-family: calibri;

            font-size: 20;

            padding: 5px;

        }


        div#heading {

            text-decoration: bold;

            text-align: center;

            margin-top: 80px;
            width: 500px;
            border: 1px solid #CEE2FA;
            text-align: center;
            color: #08438E;
            background-color: #CEE2FA;
            font-family: calibri;
            font-size: 20;
            padding: 5px;

        }

        h4 {
            padding: 0;
            margin: 0;
        }
    </style>
</head>

<body>
    <center>
        <div id="heading">
            <b>Theatre Drama</b>
        </div>
        <div id="maincontent">
        <h4>Your Ticket Details:</h4>
       <p><b>Movie Name:</b><span id="movie-name"></span></p>
    <p><b>Starring:</b> <span id="starring"></span></p>
    <p><b>Language:</b> <span id="language"></span></p>
    <p><b>Rating:</b> <span id="rating"></span></p>
        <br>
        <script>

    const movieName = "The Shawshank Redemption";
    const starring = ["Tim Robbins", "Morgan Freeman"];
    const language = "English";
    const rating = 9.3;

    // set text content of HTML elements
    document.getElementById("movie-name").textContent = movieName;
    document.getElementById("starring").textContent = starring.join(", ");
    document.getElementById("language").textContent = language;
    document.getElementById("rating").textContent = rating;
        </script>
     </div>
  </center>
</body>
</html>
```


IEKart
How t
Brows
Inaug
Terms
welco
Tic
File
E:/Lara/Materials/CSE%203-2/Mean%20Stack%20Development-R2...

Theatre Drama

Your Ticket Details:

Movie Name: The Shawshank Redemption

Starring: Tim Robbins, Morgan Freeman

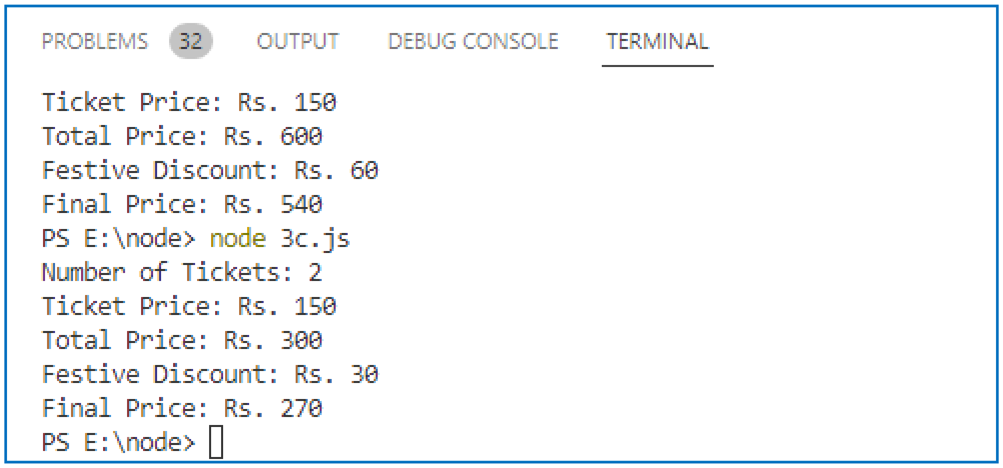
Language: English

Rating: 9.3

3c. Write JavaScript code to book movie tickets online and calculate the total price, considering the number of tickets and price per ticket as Rs. 150. Also, apply a festive season discount of 10% and calculate the discounted amount.

```
// Define the number of tickets and price per ticket
const numTickets = 2;
const ticketPrice = 150;

// Calculate the total price
let totalPrice = numTickets * ticketPrice;

// Calculate the discounted amount for festive season
let festiveDiscount = totalPrice * 0.1;

// Calculate the final price after applying the discount
let finalPrice = totalPrice - festiveDiscount;

// Output the results
console.log(`Number of Tickets: ${numTickets}`);
console.log(`Ticket Price: Rs. ${ticketPrice}`);
console.log(`Total Price: Rs. ${totalPrice}`);
console.log(`Festive Discount: Rs. ${festiveDiscount}`);
console.log(`Final Price: Rs. ${finalPrice}`);
```

Output:

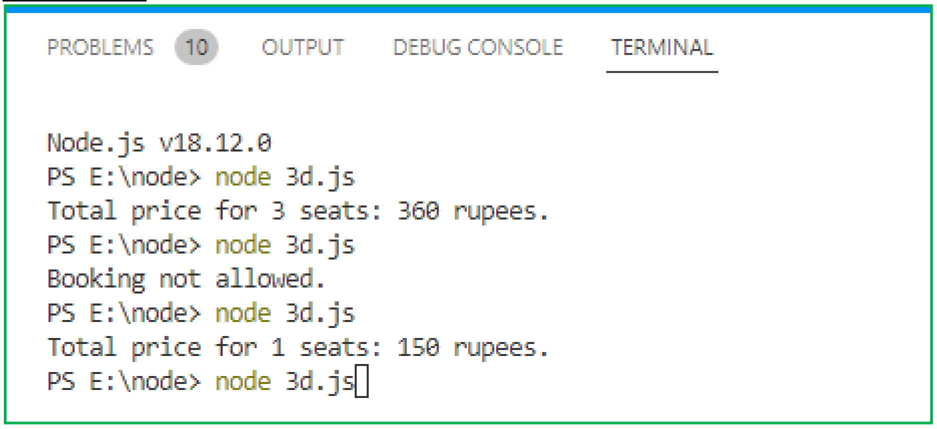


```
Ticket Price: Rs. 150
Total Price: Rs. 600
Festive Discount: Rs. 60
Final Price: Rs. 540
PS E:\node> node 3c.js
Number of Tickets: 2
Ticket Price: Rs. 150
Total Price: Rs. 300
Festive Discount: Rs. 30
Final Price: Rs. 270
PS E:\node>
```

3d. Write a JavaScript code to book movie tickets online and calculate the total price based on the 3 conditions: (a) If seats to be booked are not more than 2, the cost per ticket remains Rs. 150. (b) If seats are 6 or more, booking is not allowed. (c) If seats are between 3 to 5, the cost per ticket remains Rs.120.

```
// Assuming the number of seats to be booked is stored in a variable called
`numSeats`
const numSeats = 1; // example value

// Define the cost per ticket
var costPerTicket = 150;

// Define the maximum number of seats allowed
const maxSeatsAllowed = 5;

// Define the minimum number of seats required to make a booking
const minSeatsRequired = 1;

// Check if the number of seats is within the allowed range
if (numSeats > maxSeatsAllowed || numSeats < minSeatsRequired) {
  console.log('Booking not allowed.');
}
else{ if(numSeats>=minSeatsRequired && numSeats<=2)
{
  // Calculate the total price
  const totalPrice = numSeats * costPerTicket;
  console.log(`Total price for ${numSeats} seats: ${totalPrice} rupees.`);
}
else{
  var costPerTicket=120;
  // eslint-disable-next-line linebreak-style
  // Calculate the total price
  const totalPrice = numSeats * costPerTicket;
  console.log(`Total price for ${numSeats} seats: ${totalPrice} rupees.`);
}
}
```

Output:



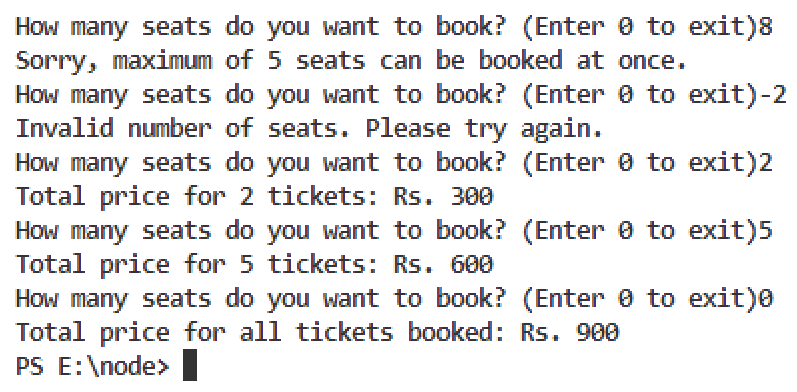
```
Node.js v18.12.0
PS E:\node> node 3d.js
Total price for 3 seats: 360 rupees.
PS E:\node> node 3d.js
Booking not allowed.
PS E:\node> node 3d.js
Total price for 1 seats: 150 rupees.
PS E:\node> node 3d.js
```

3e. Write a JavaScript code to book movie tickets online and calculate the total price based on the 3 conditions: (a) If seats to be booked are not more than 2, the cost per ticket remains Rs. 150. (b) If seats are 6 or more, booking is not allowed. (c) If tickets are in between 3, 4, 5 ticket cost is Rs.120 using Looping statements.

we can use the `readline-sync` library to get user input. `readline.questionInt` method to get an integer input from the user.

```
const readline = require('readline-sync');

const ticketPrice1 = 150; // ticket price if 1 or 2 tickets are booked
const ticketPrice2 = 120; // ticket price if 3, 4, or 5 tickets are booked
const maxSeatsAllowed = 5; // maximum seats allowed to book at once

function bookTickets() {
  let totalPrice = 0;
  let numSeats = readline.questionInt("How many seats do you want to book? (Enter
0 to exit)");

  while (numSeats !== 0) {
    if (numSeats < 1 ) {
      console.log("Invalid number of seats. Please try again.");
    }
    else if (numSeats <= 2) {
      totalPrice += numSeats * ticketPrice1;
      console.log(`Total price for ${numSeats} tickets: Rs. ${numSeats *
ticketPrice1}`);
    }
    else if (numSeats > maxSeatsAllowed) {
      console.log("Sorry, maximum of 5 seats can be booked at once.");
    }
    else {
      totalPrice += numSeats * ticketPrice2;
      console.log(`Total price for ${numSeats} tickets: Rs. ${numSeats *
ticketPrice2}`);
    }
    numSeats = readline.questionInt("How many seats do you want to book? (Enter 0
to exit)");
  }

  console.log(`Total price for all tickets booked: Rs. ${totalPrice}`);
}

// example usage
bookTickets();
```

Output:

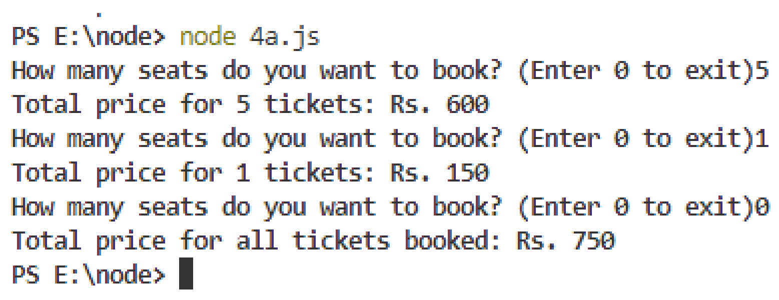


```
How many seats do you want to book? (Enter 0 to exit)8
Sorry, maximum of 5 seats can be booked at once.
How many seats do you want to book? (Enter 0 to exit)-2
Invalid number of seats. Please try again.
How many seats do you want to book? (Enter 0 to exit)2
Total price for 2 tickets: Rs. 300
How many seats do you want to book? (Enter 0 to exit)5
Total price for 5 tickets: Rs. 600
How many seats do you want to book? (Enter 0 to exit)0
Total price for all tickets booked: Rs. 900
PS E:\node>
```

4a. Write a JavaScript code to book movie tickets online and calculate the total price based on the 3 conditions: (a) If seats to be booked are not more than 2, the cost per ticket remains Rs. 150. (b) If seats are 6 or more, booking is not allowed. (c) If tickets are in between 3, 4, 5 ticket cost is Rs.120 using arrow function, scope of variables.

```
const readline = require('readline-sync');

const bookTickets = () => {
  const ticketPrice1 = 150; // ticket price if 1 or 2 tickets are booked
  const ticketPrice2 = 120; // ticket price if 3, 4, or 5 tickets are booked
  const maxSeatsAllowed = 5; // maximum seats allowed to book at once

  let totalPrice = 0;

  while (true) {
    let numSeats = readline.questionInt("How many seats do you want to book?
(Enter 0 to exit)");

    if (numSeats === 0) {
      break;
    }

    if (numSeats < 1 || numSeats > maxSeatsAllowed) {
      console.log("Invalid number of seats. Please try again.");
      continue;
    }

    let pricePerTicket = numSeats <= 2 ? ticketPrice1 : numSeats >= 6 ? 0 :
ticketPrice2;
    if (pricePerTicket === 0) {
      console.log("Sorry, maximum of 5 seats can be booked at once.");
      continue;
    }

    let totalCost = numSeats * pricePerTicket;
    totalPrice += totalCost;
    console.log(`Total price for ${numSeats} tickets: Rs. ${totalCost}`);
  }

  console.log(`Total price for all tickets booked: Rs. ${totalPrice}`);
};

bookTickets();
```

Output:

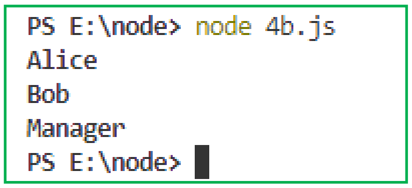


```
PS E:\node> node 4a.js
How many seats do you want to book? (Enter 0 to exit)5
Total price for 5 tickets: Rs. 600
How many seats do you want to book? (Enter 0 to exit)1
Total price for 1 tickets: Rs. 150
How many seats do you want to book? (Enter 0 to exit)0
Total price for all tickets booked: Rs. 750
PS E:\node>
```

4b. Create an Employee class extending from a base class Person. Hints: (i) Create a class Person with name and age as attributes. (ii) Add a constructor to initialize the values (iii) Create a class Employee extending Person with additional attributes role

```
class Person {
  constructor(name, age) {
    this.name = name;
    this.age = age;
  }
}

class Employee extends Person {
  constructor(name, age, role) {
    super(name, age);
    this.role = role;
  }
}
const person = new Person("Alice", 30);
const employee = new Employee("Bob", 40, "Manager");

console.log(person.name);  // "Alice"
console.log(employee.name);  // "Bob"
console.log(employee.role);  // "Manager"
```

Output:

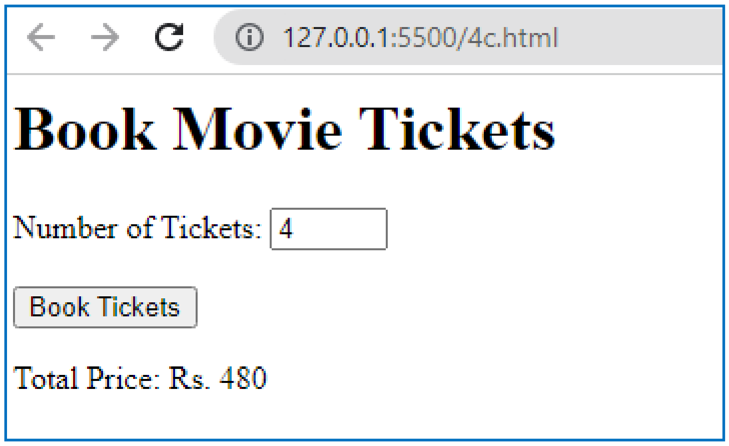


```
PS E:\node> node 4b.js
Alice
Bob
Manager
PS E:\node> █
```

4c. Write a JavaScript code to book movie tickets online and calculate the total price based on the 3 conditions: (a) If seats to be booked are not more than 2, the cost per ticket remains Rs. 150. (b) If seats are 6 or more, booking is not allowed. (c) If tickets are in between 3, 4, 5 ticket cost is Rs.120 using in-built events and handlers.

```
<html>
  <body>
    <h1>Book Movie Tickets</h1>
    <form id="bookingForm">
      <label for="numberOfTickets">Number of Tickets:</label>
      <input type="number" id="numberOfTickets" name="numberOfTickets" min="1"
max="5">
```

```
      <br><br>
      <input type="submit" value="Book Tickets">
    </form>
    <p id="totalPrice"></p>
    <script>
      const bookingForm = document.getElementById("bookingForm");
      const numberOfTickets = document.getElementById("numberOfTickets");
      const totalPrice = document.getElementById("totalPrice");

      bookingForm.addEventListener("submit", (event) => {
        event.preventDefault();  // prevent form submission

        const numberOfSeats = Number(numberOfTickets.value);

        if (numberOfSeats <= 2) {
          totalPrice.textContent = `Total Price: Rs. ${numberOfSeats * 150}`;
        } else if (numberOfSeats >= 6) {
          totalPrice.textContent = "Booking not allowed for 6 or more seats";
        } else {
          totalPrice.textContent = `Total Price: Rs. ${numberOfSeats * 120}`;
        }
      });
    </script>
  </body>
</html>
```



Book Movie Tickets

Number of Tickets: 4

Book Tickets

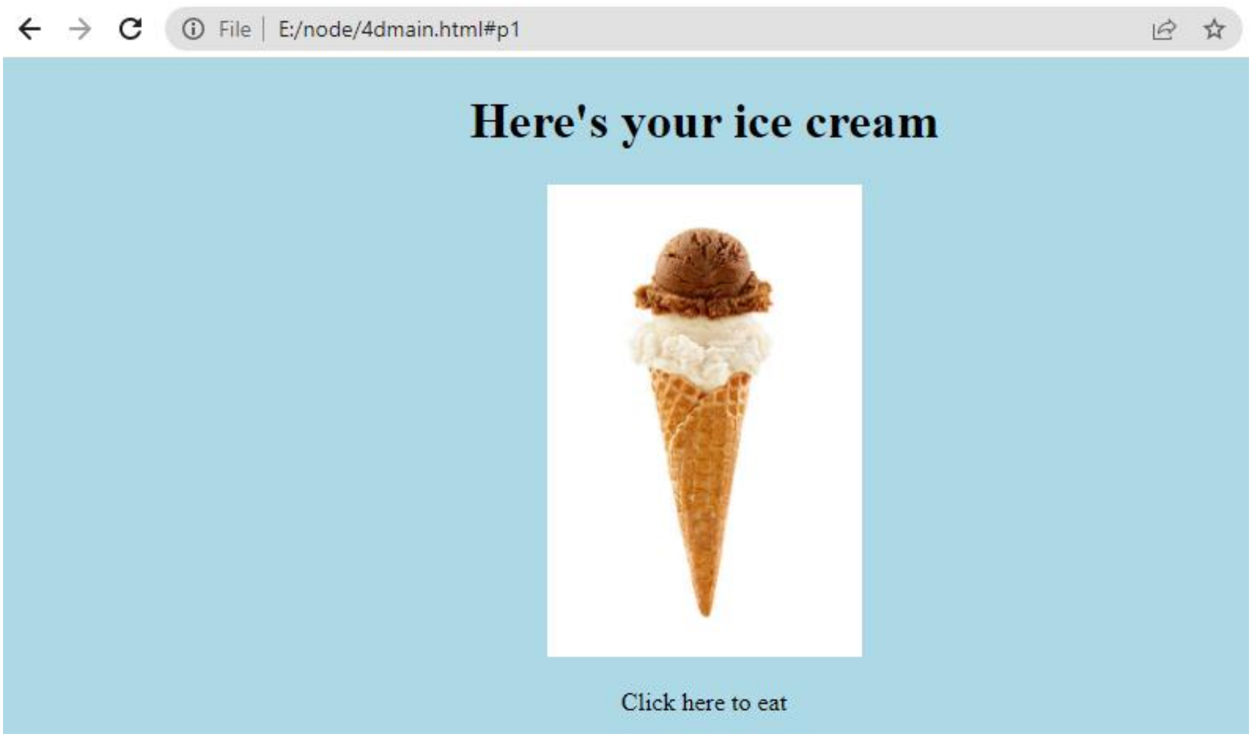
Total Price: Rs. 480

4d. If a user clicks on the given link, they should see an empty cone, a different heading, and a different message and a different background color. If user clicks again, they should see a re-filled cone, a different heading, a different message, and a different background color.

```
<!DOCTYPE html>
<html>
```

```
<head>
    <title>DOM API for HTML modification</title>
</head>

    <body style="background-color:lightblue;text-align:center">
        <h1 id="hdr1">Here's your ice cream</h1>
        <img id="img1" src="full-cone2.jpg" alt="Ice cream is not ready"
height="300" width="200" >
        <p id="p1" onclick="eat()"><a href="#p1">Click here to eat</a></p>
        <script>
           var eaten = false;

function eat() {
    if (!eaten) {
        document.getElementById("hdr1").innerHTML = "You ate the ice cream!";
        document.getElementById("img1").src = "emptycone1.jpg";
        document.getElementById("p1").innerHTML = "Click here to get another
one";
        document.body.style.backgroundColor = "pink";
        eaten = true;
    } else {
        document.getElementById("hdr1").innerHTML = "Here's your ice cream";
        document.getElementById("img1").src = "full-cone2.jpg";
        document.getElementById("p1").innerHTML = "Click here to eat";
        document.body.style.backgroundColor = "lightblue";
        eaten = false;
    }
}
        </script>
    </body>

    </html>
```

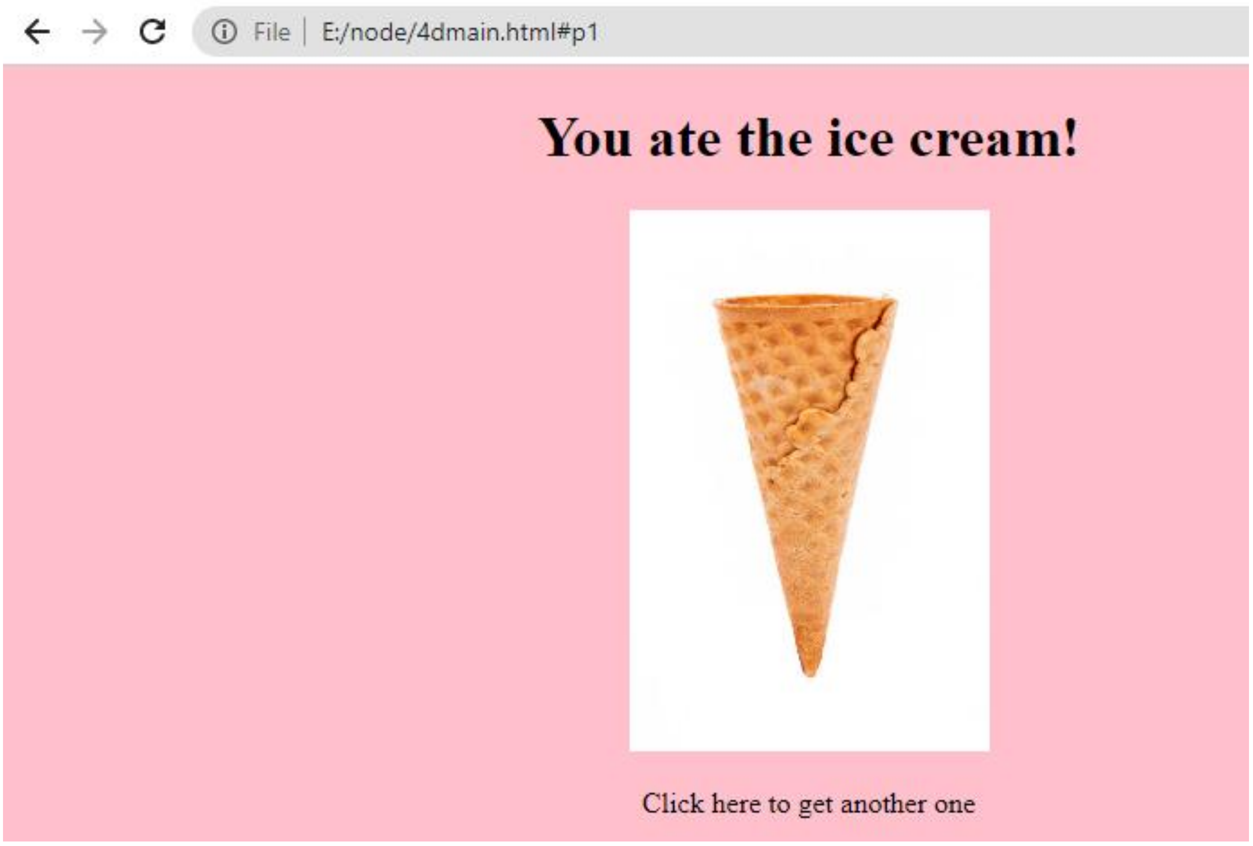
Output:



File | E:/node/4dmain.html#p1

Here's your ice cream

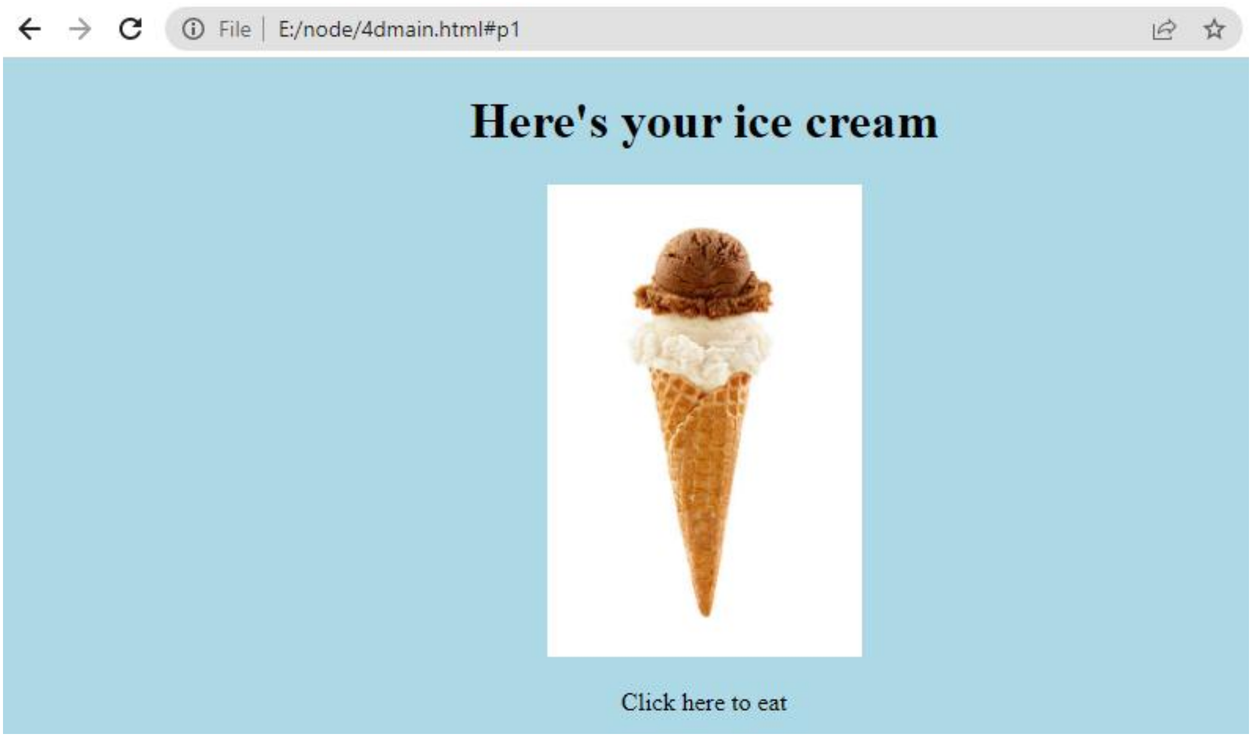
Click here to eat



File | E:/node/4dmain.html#p1

You ate the ice cream!

Click here to get another one


E:/node/4dmain.html#p1

Here's your ice cream

Click here to eat

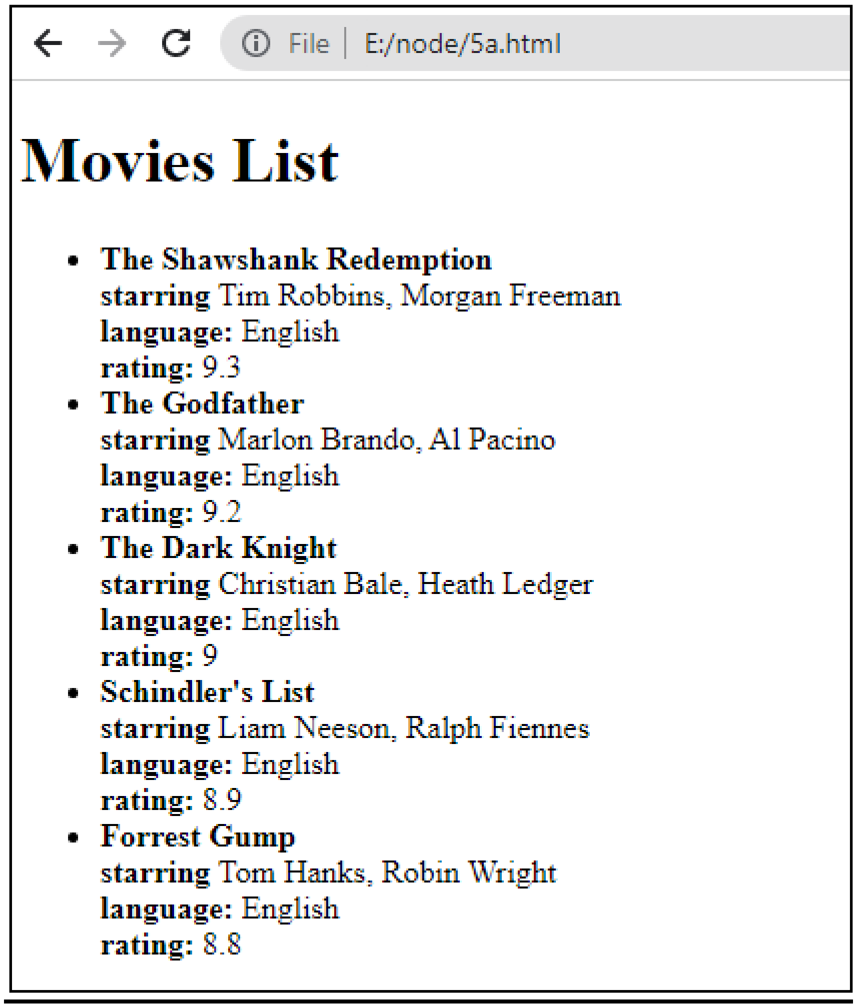
5a. Create an array of objects having movie details. The object should include the movie name, starring, language, and ratings. Render the details of movies on the page using the array.

5a.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Movies List</title>
</head>
<body>
    <h1>Movies List</h1>
    <ul id="movies-list"></ul>
    <script src="5ajs.js"></script>
</body>
</html>
```

5ajs.js

```
const movies = [
    {
        name: "The Shawshank Redemption",
        starring: "Tim Robbins, Morgan Freeman",
        language: "English",
        rating: 9.3
    },
    {
        name: "The Godfather",
        starring: "Marlon Brando, Al Pacino",
        language: "English",
        rating: 9.2
    },
    {
        name: "The Dark Knight",
        starring: "Christian Bale, Heath Ledger",
        language: "English",
        rating: 9.0
    },
    {
        name: "Schindler's List",
        starring: "Liam Neeson, Ralph Fiennes",
        language: "English",
        rating: 8.9
    },
    {
        name: "Forrest Gump",
        starring: "Tom Hanks, Robin Wright",
        language: "English",
        rating: 8.8
    }
];

const moviesList = document.getElementById("movies-list");

movies.forEach(function(movie) {
    const listItem = document.createElement("li");
    listItem.innerHTML = `<strong>${movie.name}</strong><br> <b>starring</b>
${movie.starring}<br> <b>language:</b> ${movie.language}<br> <b>rating:</b>
${movie.rating}`;
    moviesList.appendChild(listItem);
});
```
Output:


File | E:/node/5a.html

Movies List

- **The Shawshank Redemption**
  starring Tim Robbins, Morgan Freeman
  language: English
  rating: 9.3
- **The Godfather**
  starring Marlon Brando, Al Pacino
  language: English
  rating: 9.2
- **The Dark Knight**
  starring Christian Bale, Heath Ledger
  language: English
  rating: 9
- **Schindler's List**
  starring Liam Neeson, Ralph Fiennes
  language: English
  rating: 8.9
- **Forrest Gump**
  starring Tom Hanks, Robin Wright
  language: English
  rating: 8.8

5b. Simulating a periodic stock price change and displaying on the console.

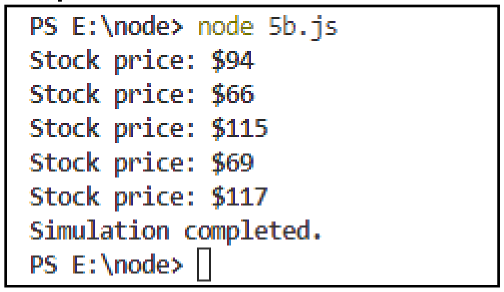
Hints:

- Create a method which returns a random number - use Math.random, floor and other methods to return a rounded value.
  - Invoke the method for every three seconds and stop when the count is 5 – use the setInterval method.
  - Since setInterval is an async method, enclose the code in a Promise and handle the response generated in a success callback.
  - The random value returned from the method every time can be used as a stock price and displayed on the console.

```
function getRandomPrice() {
  // Generate a random number between 50 and 150
  return Math.floor(Math.random() * 101) + 50;
}

function simulateStockPrice() {
  return new Promise((resolve, reject) => {
    let count = 0;
```

```
      const intervalId = setInterval(() => {
        if (count < 5) {
          const price = getRandomPrice();
          console.log(`Stock price: $${price}`);
          count++;
        } else {
          clearInterval(intervalId);
          resolve('Simulation completed.');
        }
      }, 3000);
    });
  }

  simulateStockPrice()
    .then((response) => {
      console.log(response);
    })
    .catch((error) => {
      console.error(error);
    });
```

Output:



```
PS E:\node> node 5b.js
Stock price: $94
Stock price: $66
Stock price: $115
Stock price: $69
Stock price: $117
Simulation completed.
PS E:\node>
```

5c. Create a login module to validate the user.

Hints:

- Create a file login.js with a User class.
    - Create a validate method with username and password as arguments.
    - If the username and password are equal it will return "Login Successful" else will return "Unauthorized access".
- Create an index.html file with textboxes username and password and a submit button.
- Add a script tag in HTML to include index.js file.
- Create an index.js file which imports login module and invokes validate method of User class.
- On submit of the button in HTML the validate method of the User class should be invoked.

- Implement the validate method to send the username and password details entered by the user and capture the return value to display in the alert.

`login.mjs`

login.mjs

5c.js

5c.html

6a. Verify how to execute different functions successfully in the Node.js platform

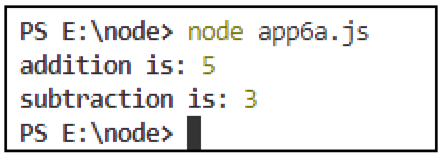
functions.js

```
function add(a, b) {
  return a + b;
}

function subtract(a, b) {
  return a - b;
}

module.exports = {
  add,
  subtract,
};
```

App6a.js

```
const { add, subtract } = require('./functions');
console.log("addition is:",add(2, 3)); // Output: 5
console.log("subtraction is:",subtract(5, 2)); // Output: 3
```

Output:



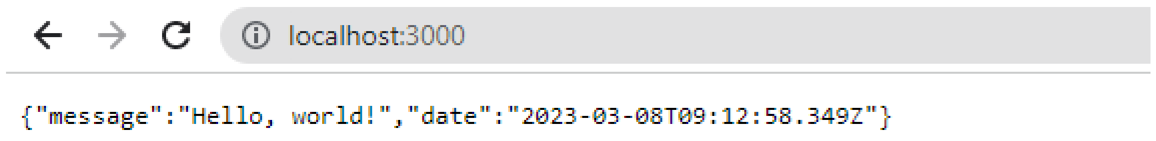
```
PS E:\node> node app6a.js
addition is: 5
subtraction is: 3
PS E:\node>
```

6b. Write a program to show the workflow of JavaScript code executable by creating web server in Node.js.

```
// Import the http module to create a web server
```

```
const http = require('http');

// Create the server
const server = http.createServer((req, res) => {
  // Set the response header to indicate that the response is in JSON format
  res.setHeader('Content-Type', 'application/json');

  // Define a JavaScript object
  const data = {
    message: 'Hello, world!',
    date: new Date()
  };

  // Convert the JavaScript object to a JSON string and send it as the response
  res.end(JSON.stringify(data));
});

// Start the server on port 3000
server.listen(3000, () => {
  console.log('Server listening on port 3000');
});
```

Output:



```
{"message":"Hello, world!","date":"2023-03-08T09:12:58.349Z"}
```

6c. Write a Node.js module to show the workflow of Modularization of Node application

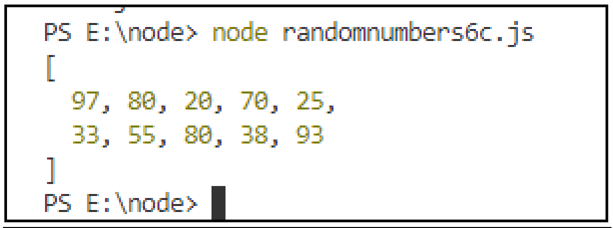
Node6c.js

```
// Define a function that generates a random number between two given numbers
function getRandomNumber(min, max) {
  return Math.floor(Math.random() * (max - min + 1) + min);
}

// Define a function that generates an array of random numbers
function generateRandomNumbers(count, min, max) {
  const numbers = [];

  for (let i = 0; i < count; i++) {
    numbers.push(getRandomNumber(min, max));
  }

  return numbers;
}
// Export the generateRandomNumbers function so it can be used in other modules
module.exports = {
  generateRandomNumbers
};
```

Randomnumbers6c.js

```
// Import the module
const randomNumbers = require('./node6c');

// Generate an array of 10 random numbers between 1 and 100
const numbers = randomNumbers.generateRandomNumbers(10, 1, 100);

// Log the array to the console
console.log(numbers);
```

Output:



```
PS E:\node> node randomnumbers6c.js
[
  97, 80, 20, 70, 25,
  33, 55, 80, 38, 93
]
PS E:\node>
```

6d. Write a program to show the workflow of restarting a Node application.

Here's how the program works:

1. A **`logMessage`** function is defined that logs a message to the console.
2. The **`setInterval`** function is used to call the **`logMessage`** function every 5 seconds. The **`setInterval`** function returns an interval ID that can be used to stop the interval later.
3. The **`process.on`** method is used to listen for the **`SIGTERM`** signal. The **`SIGTERM`** signal is sent to the process to request that it terminate gracefully.
4. When the **`SIGTERM`** signal is received, the program logs a message to the console and stops the interval by calling the **`clearInterval`** function with the interval ID.
5. The program then restarts the application by calling the **`main`** function again.

6. The **`main`** function logs a message to the console and starts the interval again by calling **`setInterval`** with the **`logMessage`** function.
7. The **`main`** function is called at the end of the program to start the application.

To test the restarting functionality of this program, you can run it in a terminal and then send the **`SIGTERM`** signal to the process. On Linux and macOS, you can do this by pressing **`Ctrl-C`** in the terminal where the program is running. On Windows, you can use the **`taskkill`** command to send the **`SIGTERM`** signal to the process. When the **`SIGTERM`** signal is received, the program should log a message to the console and restart the application.

You can run the **`taskkill`** command from the command prompt or PowerShell on Windows. Here's an example of how you could use the **`taskkill`** command to send a **`SIGTERM`** signal to a Node.js application with a specific process ID (PID):

1. Open a command prompt or PowerShell window.
2. Run the **`tasklist`** command to list all running processes and their PIDs. Find the PID of the Node.js process that you want to stop.
3. Run the **`taskkill /PID <pid> /F`** command, replacing **`<pid>`** with the PID of the Node.js process. The **`/F`** flag is used to forcefully terminate the process.

For example, if the PID of the Node.js process is **`1234`**, you could run the following command to stop the process:

```
taskkill /PID 1234 /F
```

Note that forcefully terminating a process can cause data loss or other issues, so it's generally better to use a graceful shutdown mechanism like the one demonstrated in the previous program if possible.

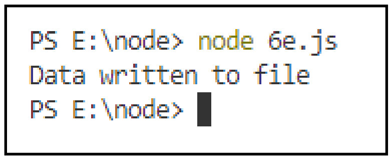
```
// Define a function that logs a message to the console
function logMessage() {
  console.log('Application is running...');
}

// Call the logMessage function every 5 seconds
const intervalId = setInterval(logMessage, 5000);
```

```
// Listen for the SIGTERM signal and restart the application when it is received
process.on('SIGTERM', () => {
  console.log('Received SIGTERM signal. Restarting application...');

  // Clear the interval so the logMessage function stops being called
  clearInterval(intervalId);

  // Restart the application by calling the main function again
  main();
});

// Define the main function
function main() {
  console.log('Starting application...');

  // Call the logMessage function every 5 seconds
  const intervalId = setInterval(logMessage, 5000);
}

// Call the main function to start the application
main();
```

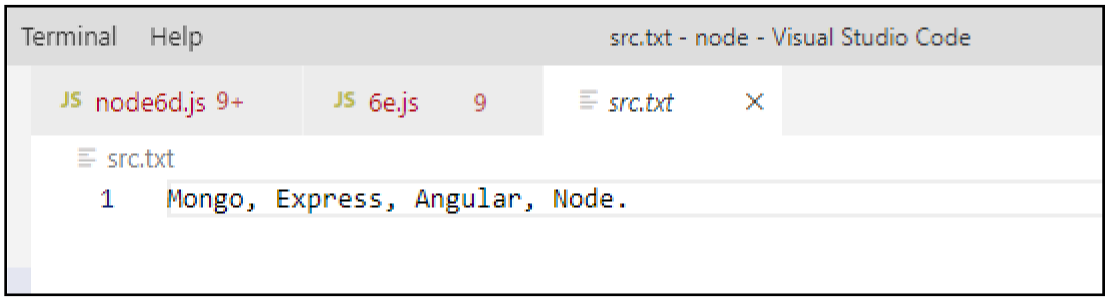
6e. Create a text file src.txt and add the following data to it. Mongo, Express, Angular, Node.

```
const fs = require('fs');

const data = "Mongo, Express, Angular, Node.";

fs.writeFile('src.txt', data, (err) => {
  if (err) throw err;
  console.log('Data written to file');
});
```

Output:



```
PS E:\node> node 6e.js
Data written to file
PS E:\node>
```



src.txt

```
Mongo, Express, Angular, Node.
```

7a. Implement routing for the AdventureTrails application by embedding the necessary code in the routes/route.js file.

you can create a routes folders, in that save file with name route.js and add the following code:

route.js

```
const express = require('express');
const router = express.Router();

// Define routes for AdventureTrails application
router.get('/', (req, res) => {
  res.send('Welcome to AdventureTrails!');
});

router.get('/trails', (req, res) => {
  res.send('Here are the trails!');
});

router.get('/trails/:id', (req, res) => {
  const id = req.params.id;
  res.send(`Here is trail ${id}!`);
});

module.exports = router;
```

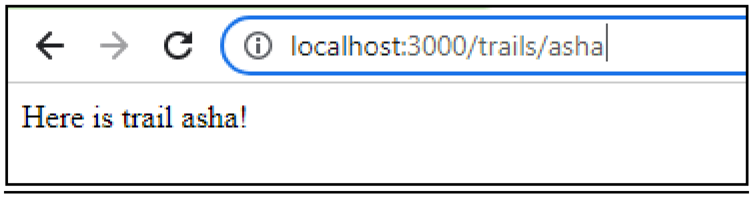
out of routes folder, take a new file and copy the code:

app7a.js

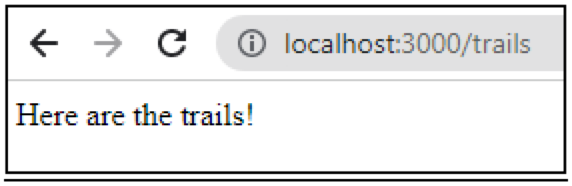
```
const express = require('express');
const app7a = express();
const routes = require('./routes/route.js');

app7a.use('/', routes);
app.get('/users/:id', (req, res) => {
    const userId = req.params.id;
```

```
  });

// Start the server
app7a.listen(3000, () => {
  console.log('AdventureTrails server started on port 3000');
});
```

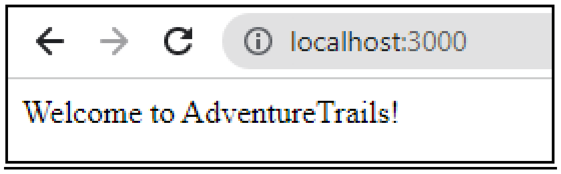
Output:



```
localhost:3000/trails/asha

Here is trail asha!
```



```
localhost:3000/trails

Here are the trails!
```



```
localhost:3000

Welcome to AdventureTrails!
```

7b. In myNotes application: (i) we want to handle POST submissions. (ii) display customized error messages. (iii) perform logging.

1. To handle POST submissions in your Express.js application, you need to use the **`body-parser`** middleware. This middleware parses the request body and makes it available in **`req.body`**. To use **`body-parser`**, first install it using npm:

   npm install body-parser

   Then, require it in your app.js file and use it as middleware:

```
const bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: false }));
app.use(bodyParser.json());
```

2. To display customized error messages, you can use Express.js's built-in error handling middleware. To define custom error messages, create a middleware function that takes four parameters: **`err`**, **`req`**, **`res`**, and

next. You can then use the **res.status()** and **res.send()** methods to send an appropriate error message to the client. For example:

```
app.use((err, req, res, next) => {
    console.error(err.stack);
    res.status(500).send('Oops! Something went wrong.');
  });
```

This middleware will catch any errors that occur during the request handling process and send an error message to the client.

3. To perform logging, you can use the **morgan** middleware. This middleware logs incoming requests and their associated responses to the console. To use **morgan**, first install it using npm:

npm install morgan

Then, require it in your app.js file and use it as middleware:

```
const morgan = require('morgan');
app.use(morgan('combined'));
```

This will log incoming requests and their associated responses to the console in the combined log format.

With these steps, you can handle POST submissions, display customized error messages, and perform logging in your Express.js application.

To save and execute TypeScript programs, you will need to follow these steps:

1. Install the TypeScript compiler (if you haven't already): You can install the TypeScript compiler globally on your machine using npm (Node Package Manager) by running the following command in your terminal: **`npm install -g typescript`**.
2. Create a new TypeScript file: Create a new file with a **`.ts`** extension in your preferred code editor.
3. Write TypeScript code: Write your TypeScript code in the **`.ts`** file.
4. Compile the TypeScript code: Once you've written your TypeScript code, you need to compile it into JavaScript using the TypeScript compiler. You can do this by running the **`tsc`** command followed by the name of your TypeScript file in your terminal. For example: **`tsc myfile.ts`**.
5. Execute the JavaScript file: After compiling the TypeScript code into JavaScript, you can execute it using Node.js. Run the **`node`** command followed by the name of the JavaScript file in your terminal. For example: **`node myfile.js`**.

Alternatively, you can use a tool like **`ts-node`** to directly execute TypeScript files without having to manually compile them first. You can install **`ts-node`** globally using npm and then run your TypeScript files using the **`ts-node`** command followed by the name of your TypeScript file. For example: **`ts-node myfile.ts`**.

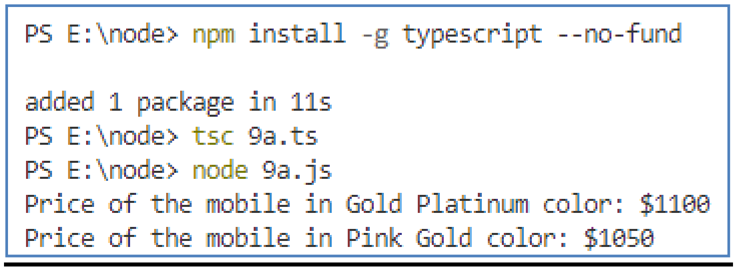
Keep in mind that TypeScript code can also be executed in a browser, but the setup may differ depending on your development environment and project requirements.

9a. On the page, display the price of the mobile-based in three different colors. Instead of using the number in our code, represent them by string values like GoldPlatinum, PinkGold, SilverTitanium.

9a.ts

```
const GoldPlatinum: string = "1100";
const PinkGold: string = "1050";
const SilverTitanium: string = "1000";

console.log(`Price of the mobile in Gold Platinum color: $${GoldPlatinum}`);
console.log(`Price of the mobile in Pink Gold color: $${PinkGold}`);
console.log(`Price of the mobile in Silver Titanium color: $${SilverTitanium}`);
```

Output:



```
PS E:\node> npm install -g typescript --no-fund

added 1 package in 11s
PS E:\node> tsc 9a.ts
PS E:\node> node 9a.js
Price of the mobile in Gold Platinum color: $1100
Price of the mobile in Pink Gold color: $1050
```

9b. Define an arrow function inside the event handler to filter the product array with the selected product object using the productId received by the function. Pass the selected product object to the next screen.

```
import type { Product } from './types';
interface Product {
  productId: number;
  productName: string;
  productPrice: number;
}

const products: Product[] = [
  { productId: 1, productName: "Product 1", productPrice: 10 },
  { productId: 2, productName: "Product 2", productPrice: 20 },
  { productId: 3, productName: "Product 3", productPrice: 30 }
];
function handleProductClick(productId: number) {
  const selectedProduct: Product | undefined = products.find(product =>
product.productId === productId);

  if (selectedProduct) {
    // Navigate to the next screen with the selected product object
    navigateToNextScreen(selectedProduct);
  }
}
function navigateToNextScreen(selectedProduct: Product) {
    console.log(`Navigating to next screen with product:
${selectedProduct.productName}`);
  }
```

In this example, we define an interface named **`Product`** to represent the shape of our product objects. We then define an array of **`Product`** objects named **`products`**, which contains three sample products.

Next, we define an event handler function named **`handleProductClick`** that takes a **`productId`** parameter. Inside this function, we use the **`Array.find()`** method to filter the **`products`** array and retrieve the product object that matches the given **`productId`**. If a product is found, we pass the selected **`Product`** object to the **`navigateToNextScreen()`** function to navigate to the next screen with the selected product object.

Note that we use the arrow function syntax (**`product => product.productId === productId`**) inside the **`Array.find()`** method to define a callback function that filters the array based on the **`productId`** parameter. This arrow function takes a **`product`** parameter (representing each element in the array) and returns **`true`** if the **`productId`** of the **`product`** parameter matches the given **`productId`** argument.

Please note that **`navigateToNextScreen()`** is a placeholder function for demonstration purposes only. You should replace it with the actual function that will handle the navigation to the next screen in your application.

The error message you are seeing (**`error TS6231: Could not resolve the path 'init' with the extensions...`**) suggests that TypeScript is unable to find a module or file named **`init`** that it needs to run. This error can occur if you accidentally typed the wrong command or if the **`tsc`** command is not installed on your system.

To fix this error, you can try the following steps:

1. Double-check that you are running the correct command to generate the **`tsconfig.json`** file. The correct command is:

   csharpCopy code
   ```
   init
   ```
   Note that **`--init`** is a flag to tell the **`tsc`** command to create a new **`tsconfig.json`** file.
2. Make sure that you have the TypeScript compiler (**`tsc`**) installed on your system. You can check this by running the following command:

   cssCopy code
   ```
   --version
   ```
   If you see a version number (e.g. **`Version 4.5.4`**), then TypeScript is installed on your system. If you see an error message, you may need to install TypeScript globally using the following command:

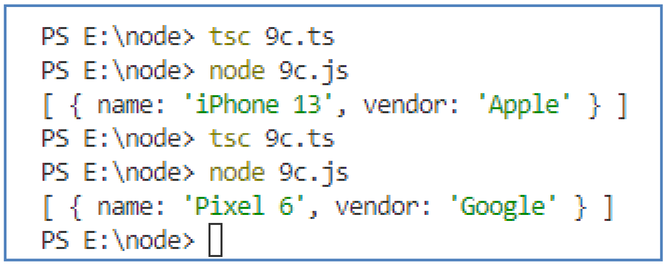
   rubyCopy code
   ```

   $
   ```
3. If you have installed TypeScript globally, you may need to specify the full path to the **`tsc`** command when running the **`--init`** command. For example:

   luaCopy code
   ```
   local
   ```
   This assumes that the **`tsc`** command is installed in the **`/usr/local/bin`** directory. You may need to adjust the path based on where you installed TypeScript on your system.
4. If none of these steps fix the error, you may need to provide more information about your environment and the exact command you are running in order to diagnose the problem.

9c. Consider that developer needs to declare a function - getMobileByVendor which accepts string as input parameter and returns the list of mobiles.

```
interface Mobile {
  name: string;
  vendor: string;
}

function getMobileByVendor(vendor: string): Mobile[] {
  // Assuming you have a list of mobiles with their respective vendors
  const mobiles: Mobile[] = [
    { name: 'iPhone 13', vendor: 'Apple' },
    { name: 'Galaxy S21', vendor: 'Samsung' },
    { name: 'Pixel 6', vendor: 'Google' },
    { name: 'Mi 11 Ultra', vendor: 'Xiaomi' },
    { name: 'OnePlus 9 Pro', vendor: 'OnePlus' },
  ];

  // Filter the mobiles based on the vendor parameter
  const filteredMobiles = mobiles.filter(
    (mobile) => mobile.vendor.toLowerCase() === vendor.toLowerCase()
  );

  return filteredMobiles;
}
const appleMobiles = getMobileByVendor('Google');
console.log(appleMobiles);
```

Output:



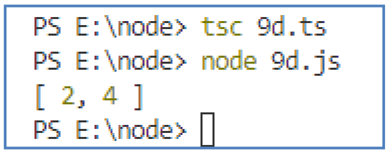
```
PS E:\node> tsc 9c.ts
PS E:\node> node 9c.js
[ { name: 'iPhone 13', vendor: 'Apple' } ]
PS E:\node> tsc 9c.ts
PS E:\node> node 9c.js
[ { name: 'Pixel 6', vendor: 'Google' } ]
PS E:\node> 
```

9d. Consider that developer needs to declare a manufacturer's array holding 4 objects with id and price as a parameter and needs to implement an arrow function - myfunction to populate the id parameter of manufacturers array whose price is greater than or equal to 200 dollars then below mentioned code-snippet would fit into this requirement.

```
interface Manufacturer {
  id: number;
  price: number;
}
```

```
const manufacturers: Manufacturer[] = [
  { id: 1, price: 150 },
  { id: 2, price: 300 },
  { id: 3, price: 75 },
  { id: 4, price: 225 },
];

const myFunction = (manufacturers: Manufacturer[]): number[] =>
  manufacturers
    .filter((manufacturer) => manufacturer.price >= 200)
    .map((manufacturer) => manufacturer.id);

const result = myFunction(manufacturers);

console.log(result); // [2, 4]
```

Output:



```
PS E:\node> tsc 9d.ts
PS E:\node> node 9d.js
[ 2, 4 ]
PS E:\node> 
```

In this example, we start by defining an interface **`Manufacturer`** that defines the shape of the objects in the **`manufacturers`** array, which holds 4 objects with **`id`** and **`price`** as properties. We then define the **`myFunction`** arrow function, which takes in the **`manufacturers`** array as a parameter.

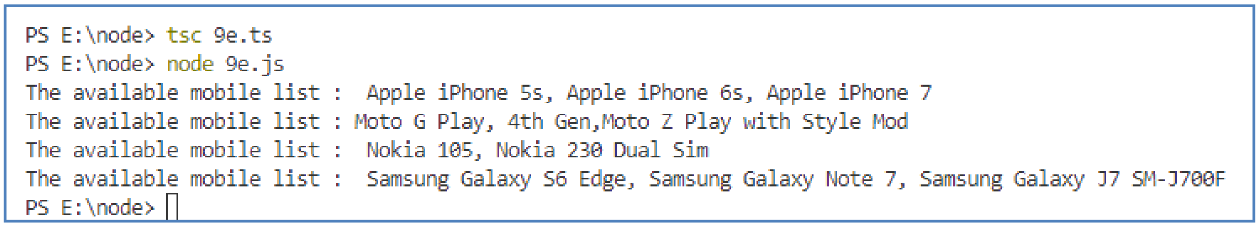
Inside **`myFunction`**, we use the **`filter`** method to filter the **`manufacturers`** array and only keep the objects where the **`price`** property is greater than or equal to 200 dollars. We then use the **`map`** method to extract the **`id`** property from each of the filtered objects and return an array of **`id`** values.

Finally, we call **`myFunction`** with the **`manufacturers`** array as an argument and log the result to the console, which should be an array containing the **`id`** values of the manufacturers whose **`price`** is greater than or equal to 200 dollars, in this case **`[2, 4]`**.

9e. Consider that developer needs to declare a function - getMobileByManufacturer with two parameters namely manufacturer and id, where manufacturer value should passed as Samsung and id parameter should be optional while invoking the function, if id is passed as 101 then this function should return Moto mobile list and if manufacturer parameter is either Samsung/Apple then this function should return respective mobile list and similar to make Samsung as default Manufacturer. Below mentioned code-snippet would fit into this requirement.

```
function getMobileByManufacturer(manufacturer: string, id?: number): string[] {
  let mobileList: string[];

  // logic to be evaluated if id parameter while invoking above declared function
  if (id) {
    if (id === 101) {
      mobileList = ['Moto G Play, 4th Gen', 'Moto Z Play with Style Mod'];
      return mobileList;
    }
  }

  // logic to return mobileList based on manufacturer category
  if (manufacturer === 'Samsung') {
    mobileList = [' Samsung Galaxy S6 Edge', ' Samsung Galaxy Note 7', ' Samsung
Galaxy J7 SM-J700F'];
    return mobileList;
  } else if (manufacturer === 'Apple') {
    mobileList = [' Apple iPhone 5s', ' Apple iPhone 6s', ' Apple iPhone 7'];
    return mobileList;
  } else {
    mobileList = [' Nokia 105', ' Nokia 230 Dual Sim'];
    return mobileList;
  }
}

// statement to invoke function with optional parameter
console.log('The available mobile list : ' + getMobileByManufacturer('Apple'));

// statement to invoke function with default parameter
console.log('The available mobile list : ' + getMobileByManufacturer(undefined,
101));

console.log('The available mobile list : ' + getMobileByManufacturer(undefined));

console.log('The available mobile list : ' + getMobileByManufacturer('Samsung',
102));
```

Output:



```
PS E:\node> tsc 9e.ts
PS E:\node> node 9e.js
The available mobile list :  Apple iPhone 5s, Apple iPhone 6s, Apple iPhone 7
The available mobile list : Moto G Play, 4th Gen,Moto Z Play with Style Mod
The available mobile list :  Nokia 105, Nokia 230 Dual Sim
The available mobile list :  Samsung Galaxy S6 Edge, Samsung Galaxy Note 7, Samsung Galaxy J7 SM-J700F
PS E:\node> 
```

In the updated function declaration on line 1, we remove the default value for the **`manufacturer`** parameter and update the type of the **`id`** parameter to be optional using the **`?`** syntax.

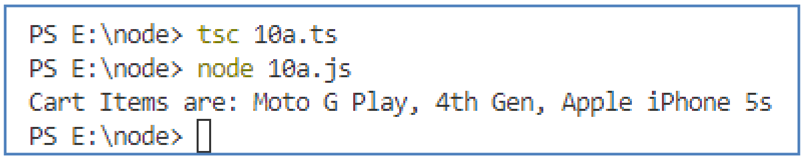
In the **`console.log`** statements on lines 19-22, we update the function calls to pass **`undefined`** as the first argument to use the default value for the **`manufacturer`** parameter, and pass **`101`** and **`102`** respectively as the **`id`** argument.

With these updates, the function should work as expected with optional parameters and default values.

10a. Implement business logic for adding multiple Product values into a cart variable which is type of string array.

```
// declaring a empty string array
const cart: string[]= [];

// arrow function logic to push values into cart array
const pushtoCart = (item: string) => { cart.push(item); };

// logic to add items into cart
function addtoCart(...productName: string[]): string[] {

    for (const item of productName) {
        pushtoCart(item);
    }
    return cart;

}

// to populate value on console
console.log('Cart Items are:' + addtoCart(' Moto G Play, 4th Gen', ' Apple iPhone 
5s'));
```

Output:

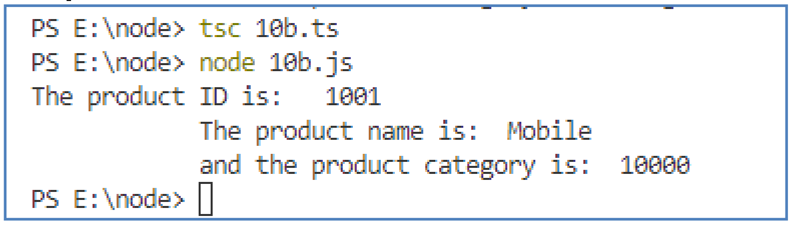


```
PS E:\node> tsc 10a.ts
PS E:\node> node 10a.js
Cart Items are: Moto G Play, 4th Gen, Apple iPhone 5s
PS E:\node> 
```

10b. Declare an interface named - Product with two properties like productId and productName with a number and string datatype and need to implement logic to populate the Product details. Add productPrice property and observe output.

```
// declaring an interface
interface Product {
    productId: number;
    productName: string;
    productPrice: number; // added property
}

// logic to display the Product details with interface object as parameter
function getProductDetails(productobj: Product): string {
    return `The product ID is:   ${productobj.productId}
            The product name is:  ${productobj.productName}
            and the product category is:  ${productobj.productPrice}`; //
modified return statement
}

// declaring a variable with interface properties
const prodObject: Product = {productId: 1001, productName: 'Mobile',
productPrice: 10000};

// invoking Product details function and logging result to console
console.log(getProductDetails(prodObject));
```

Output:

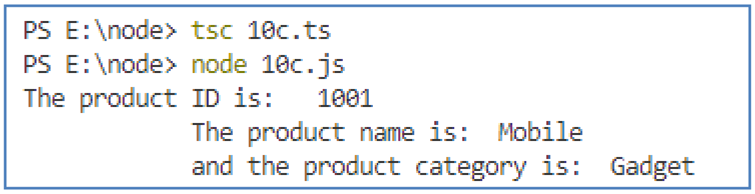


```
PS E:\node> tsc 10b.ts
PS E:\node> node 10b.js
The product ID is:   1001
            The product name is:  Mobile
            and the product category is:  10000
PS E:\node> 
```

10C. Declare an interface named - Product with two properties like productId and productName with a number and string datatype and need to implement logic to populate the Product details. Add productCategory property in interface and observe the output.

```
// declaring an interface
interface Product {
    productId: number;
    productName: string;
    productCategory: string; // added property
}

// logic to display the Product details with interface object as parameter
function getProductDetails(productobj: Product): string {
    return `The product ID is:   ${productobj.productId}
            The product name is:  ${productobj.productName}
            and the product category is:  ${productobj.productCategory}`; //
modified return statement
}

// declaring a variable with interface properties
const prodObject: Product = {productId: 1001, productName: 'Mobile',
productCategory: 'Gadget'};

// invoking Product details function and logging result to console
console.log(getProductDetails(prodObject));
```

Output:



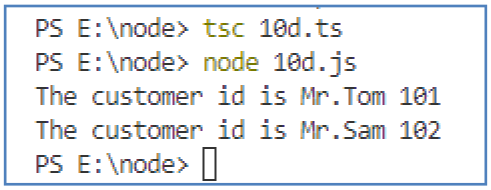
```
PS E:\node> tsc 10c.ts
PS E:\node> node 10c.js
The product ID is:   1001
            The product name is:  Mobile
            and the product category is:  Gadget
```

10d. Declare an interface with function type and access its value.

```
// declaring a function
function CreateCustomerID(name: string, id: number): string {
    return 'The customer id is ' + name + ' ' + id;
}

// declaring an interface with function type
interface StringGenerator {
    (chars: string, nums: number): string;
}

// creating a reference variable of the above declared interface
```

```
let IdGenerator: StringGenerator;

// assignment of function to interface reference variable
IdGenerator = CreateCustomerID;

// declaring a string parameter to hold return value of function type interface
const customerId: string = IdGenerator('Mr.Tom', 101);

// declaring a string parameter to hold return value of function type interface
let id2: string = IdGenerator('Mr.Sam', 102);

// line to populate the Customer Details
console.log(customerId);

// line to populate the second customer details
console.log(id2);
```

Output:



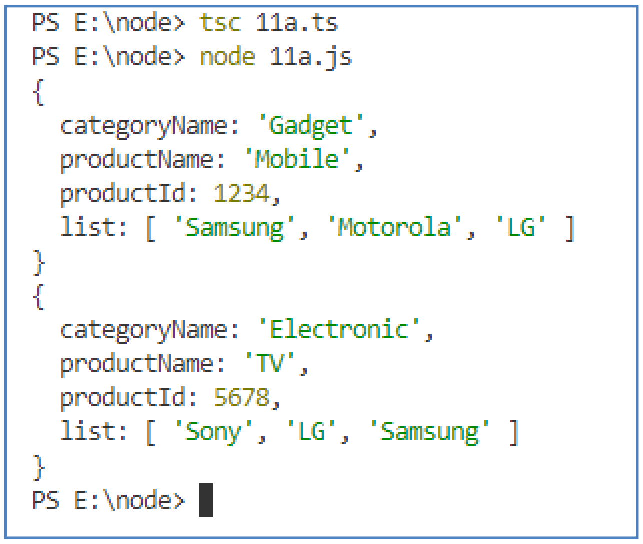
```
PS E:\node> tsc 10d.ts
PS E:\node> node 10d.js
The customer id is Mr.Tom 101
The customer id is Mr.Sam 102
PS E:\node> 
```

11a. Declare a productList interface which extends properties from two other declared interfaces like Category,Product as well as implementation to create a variable of this interface type.

```
// declaring a Category interface
interface Category {
    categoryName: string;
}

// declaring a Product interface
interface Product {
    productName: string;
    productId: number;
}

// declaring a ProductList interface which extends from Category and Product
interfaces
interface ProductList extends Category, Product {
    list: Array<string>;
}
```

```
// declaring a variable which is of type ProductList interface
const productDetails: ProductList = {
    categoryName: 'Gadget',
    productName: 'Mobile',
    productId: 1234,
    list: ['Samsung', 'Motorola', 'LG']
};

// declaring one more variable of type ProductList interface
const anotherProductDetails: ProductList = {
    categoryName: 'Electronic',
    productName: 'TV',
    productId: 5678,
    list: ['Sony', 'LG', 'Samsung']
};

// line to populate the created product variable on console
console.log(productDetails);
console.log(anotherProductDetails);
```

Output:



```
PS E:\node> tsc 11a.ts
PS E:\node> node 11a.js
{
  categoryName: 'Gadget',
  productName: 'Mobile',
  productId: 1234,
  list: [ 'Samsung', 'Motorola', 'LG' ]
}
{
  categoryName: 'Electronic',
  productName: 'TV',
  productId: 5678,
  list: [ 'Sony', 'LG', 'Samsung' ]
}
PS E:\node>
```

11b. Consider the Mobile Cart application, Create objects of the Product class and place them into the productlist array.

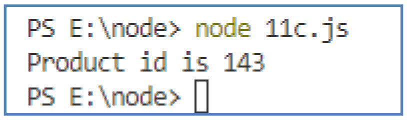
```
class Product {
  constructor(
    public id: number,
    public name: string,
    public price: number,
    public description: string,
    public image: string
  ) {}
}

let productList: Product[] = [
  new Product(
    1,
    'Samsung Galaxy S21',
    799,
    'A high-end Android smartphone with a 6.2-inch display and 5G connectivity.',
    'https://example.com/samsung-galaxy-s21.jpg'
  ),
  new Product(
    2,
    'Apple iPhone 13 Pro',
    999,
    'A high-end iOS smartphone with a 6.1-inch Super Retina XDR display and 5G
connectivity.',
    'https://example.com/iphone-13-pro.jpg'
  ),
  new Product(
    3,
    'Google Pixel 6',
    599,
    'A mid-range Android smartphone with a 6.4-inch display and 5G
connectivity.',
    'https://example.com/google-pixel-6.jpg'
  ),
];
```

11c. Declare a class named - Product with the below-mentioned declarations: (i) productId as number property (ii) Constructor to initialize this value (iii) getProductId method to return the message "Product id is <<id value>>".

```
class Product {
  private productId: number;

  constructor(productId: number) {
    this.productId = productId;
  }

  getProductId(): string {
    return `Product id is ${this.productId}`;
  }
}
const product = new Product(143);
```

```
console.log(product.getProductId()); // Output: Product id is 123
```

Output:

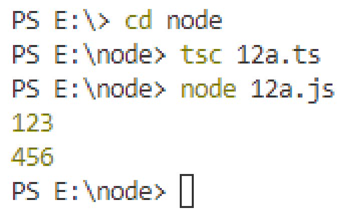


```
PS E:\node> node 11c.js
Product id is 143
PS E:\node> 
```

12a. Create a Product class with 4 properties namely productId and methods to setProductId() and getProductId().

```
class Product {
  private _productId: number;

  constructor(productId: number) {
    this._productId = productId;
  }

  public set productId(productId: number) {
    this._productId = productId;
  }

  public get productId(): number {
    return this._productId;
  }
}
const product = new Product(123);
console.log(product.productId); // Output: 123

product.productId = 456;
console.log(product.productId); // Output: 456
```

Output:



```
PS E:\> cd node
PS E:\node> tsc 12a.ts
PS E:\node> node 12a.js
123
456
PS E:\node> 
```

12b. Create a namespace called ProductUtility and place the Product class definition in it. Import the Product class inside productlist file and use it.

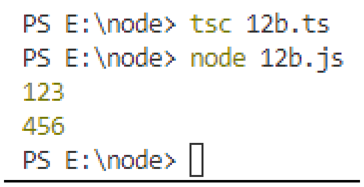
<u>Product.ts</u>

```
export namespace ProductUtility {
  export class Product {
    private _productId: number;

    constructor(productId: number) {
      this._productId = productId;
    }

    public set productId(productId: number) {
      this._productId = productId;
    }

    public get productId(): number {
      return this._productId;
    }
  }
}
```

12b.ts

```
import { ProductUtility } from './product';

const product1 = new ProductUtility.Product(123);
console.log(product1.productId); // Output: 123

const product2 = new ProductUtility.Product(456);
console.log(product2.productId); // Output: 456
```

Output:



```
PS E:\node> tsc 12b.ts
PS E:\node> node 12b.js
123
456
PS E:\node> 
```

12c. Consider the Mobile Cart application which is designed as part of the functions in a module to calculate the total price of the product using the quantity and price values and assign it to a totalPrice variable.

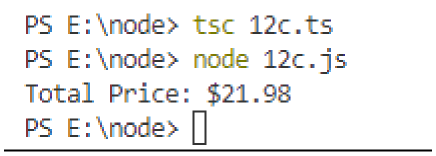
mobile-cart.ts

```
export module MobileCart {
  export function calculateTotalPrice(quantity: number, price: number): number {
    return quantity * price;
  }
}
```

12c.ts

```
import { MobileCart } from './mobile-cart';

const quantity = 2;
const price = 10.99;

const totalPrice = MobileCart.calculateTotalPrice(quantity, price);
console.log(`Total Price: $${totalPrice.toFixed(2)}`);
```

Output:

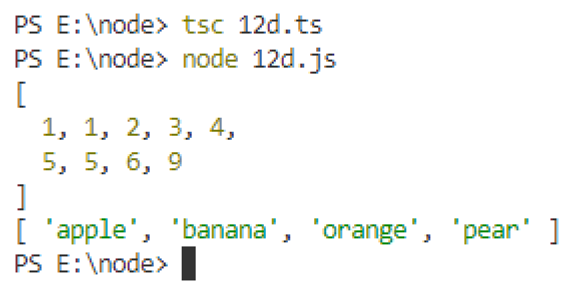


```
PS E:\node> tsc 12c.ts
PS E:\node> node 12c.js
Total Price: $21.98
PS E:\node> 
```

12d. Create a generic array and function to sort numbers as well as string values.

```
function sortArray<T>(array: T[]): T[] {
  return array.sort();
}

const numberArray: number[] = [3, 1, 4, 1, 5, 9, 2, 6, 5];
console.log(sortArray(numberArray)); // Output: [1, 1, 2, 3, 4, 5, 5, 6, 9]

const stringArray: string[] = ['banana', 'apple', 'pear', 'orange'];
console.log(sortArray(stringArray)); // Output: ['apple', 'banana', 'orange',
'pear']
```

Output:



```
PS E:\node> tsc 12d.ts
PS E:\node> node 12d.js
[
  1, 1, 2, 3, 4,
  5, 5, 6, 9
]
[ 'apple', 'banana', 'orange', 'pear' ]
PS E:\node> 
```